

Browse the Book

This chapter covers the installation and configuration of the SAP HANA system as a development environment on premise and in the cloud.

-  **"SAP HANA Development Environment"**
-  **Table of Contents**
-  **Index**
-  **The Authors**

Francesco Alborghetti, Jonas Kohlbrenner, Abani Pattanayak, Dominik Schrank, Primo Sboarina

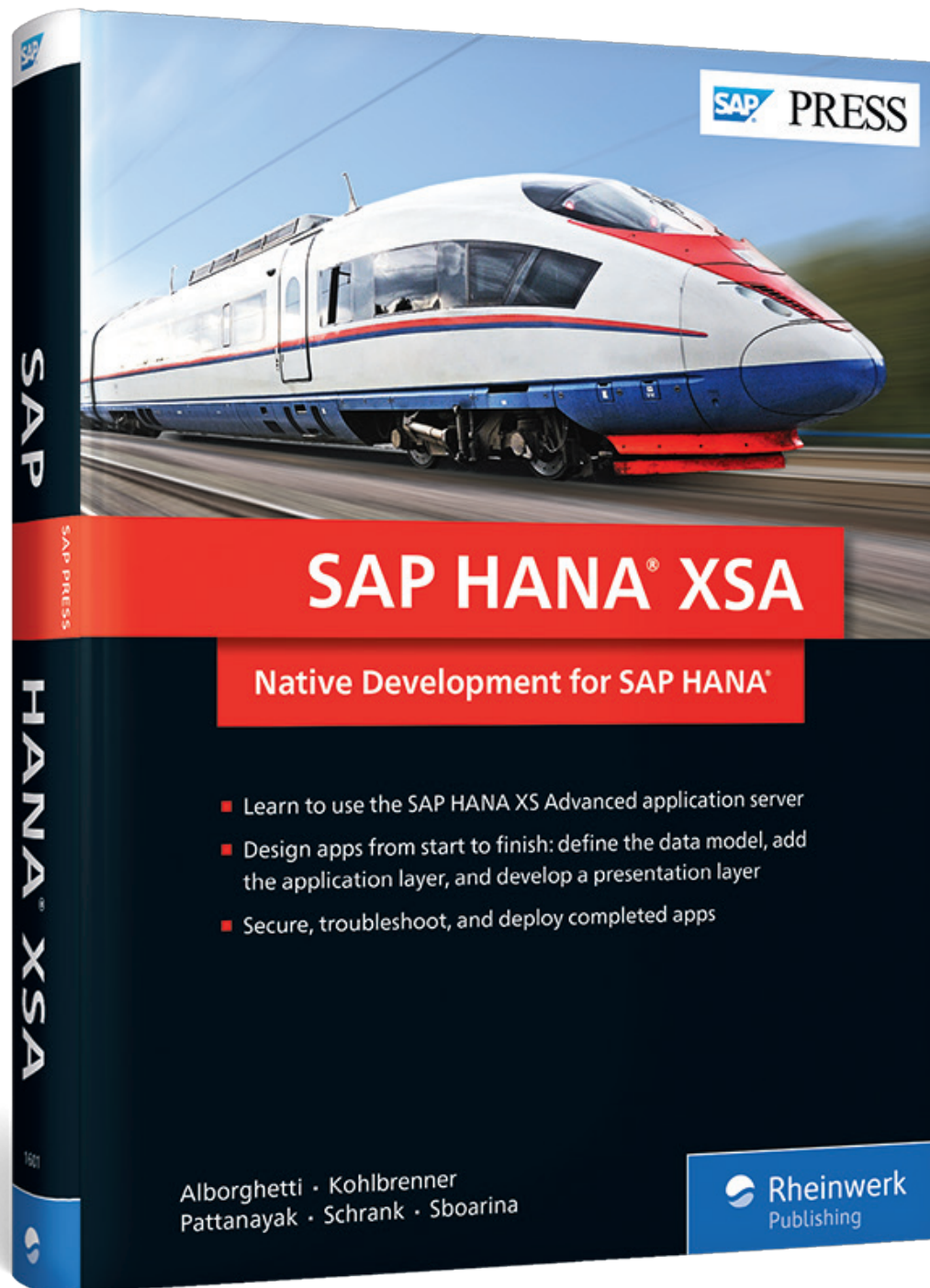
SAP HANA XSA: Native Development for SAP HANA

607 Pages, 2018, \$79.95

ISBN 978-1-4932-1601-7



www.sap-press.com/4500



Chapter 2

SAP HANA Development Environment

In this chapter, you'll learn about setting up the SAP HANA development environment in the cloud and/or on premise.

In the previous chapter, you learned the architecture and components of the SAP HANA platform. In this chapter, we'll discuss the installation and configuration of the SAP HANA system as a development environment on premise and in the cloud.

We'll explain how the SAP HANA extended application services, advanced model (SAP HANA XS Advanced) system landscape differs from the traditional SAP landscape and the option of using SAP HANA, express edition as a development system on premise and in the public cloud. We'll also discuss the options to use SAP HANA XS Advanced in the SAP Cloud Platform. Finally, we'll introduce Git as the code repository and outline the steps that organize the development environment into organizations and spaces.

2.1 System Landscape

Traditionally, SAP HANA systems are deployed with a three-tier system landscape made up of the *development* (DEV) tier, *quality* (QAS) tier, and *production* (PRD) tier. Some customers also deploy an extra *preproduction* or *user acceptance* (UAT) tier between the quality and production system. Changes/enhancements or new releases of the application follow the standard DEV-QAS-PRD or DEV-QAS-UAT-PRD transport path, depending on the deployed system landscape, as illustrated in Figure 2.1 ❶. The environments are as follows:

- **Development (DEV)**
Environment for development of new applications/releases and bug fixes and/or enhancements to the current release of application.
- **Quality (QAS)**
Environment for end-to-end functional testing and integration testing of applications.

- **Pre-production or user acceptance (UAT)**
The exact replica of the production system used for pre-production and user acceptance testing. If the customer doesn't have a separate UAT system, pre-production and user acceptance testing is supported in the QAS system.
- **Production (PRD)**
Productive system used by the customer to execute day-to-day business.

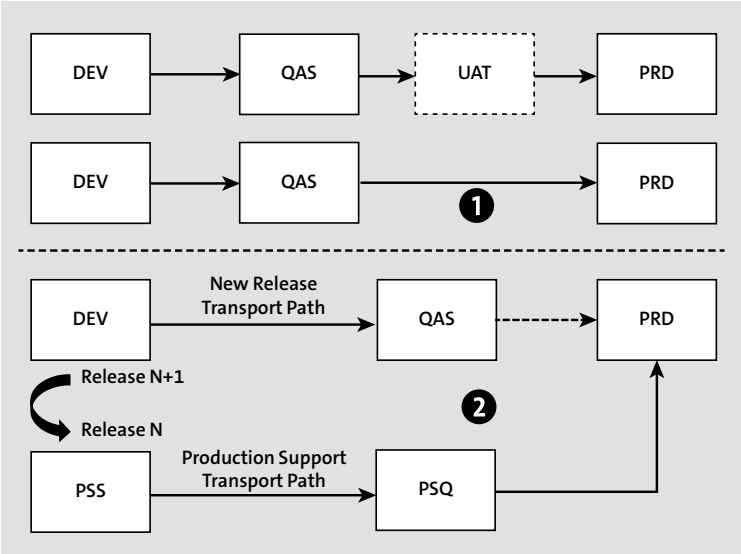


Figure 2.1 SAP N and N+1 System Landscape

Some customers may also adopt a five-tier system landscape, which is also referred to as *N and N+1 system landscape*, where N indicates the current release in production, and N+1 is the new release under development (see Figure 2.1 ②). The five-tier system landscape offers two separate transport paths, a production support path (PSS-PSQ-PRD), and a new release path (DEV-QAS-PRD). The main advantage of the N+1 landscape is that it enables the development of enhancements/new releases on one landscape while leaving the maintenance landscape free to tackle maintenance work to support day-to-day business.

The built-in SAP HANA Repository is used to keep track of new objects and object changes in the DEV or production support (PSS) systems. These change objects are used in transports to deploy applications to quality and production systems in the landscape. This has served well for application development using traditional water-fall or agile methodology in any type (three-, four-, or five-tier) landscape.

The DevOps methodology has been widely adapted in software development over the past few years to decrease long delivery cycles and to bring development (Dev) and operations (Ops) team closer. DevOps is the union of traditional development and operations as one team with common goals and objectives to facilitate *continuous integration*, *continuous delivery*, and *continuous deployment* with an emphasis on continuous process improvement, as follows:

- **Continuous integration**
Continuous integration is a development practice that requires developers to integrate code into a shared repository several times a day. Each code check-in or commit is then verified by an automated build to detect problems early.
- **Continuous delivery**
Continuous delivery is the practice that requires deployment of codes to production rapidly and safely by delivering every change to a production-like environment and ensuring business applications function as expected through automated regression testing.
- **Continuous deployment**
Continuous deployment is the practice of automatically deploying every change (that has passed the automated regression tests) to production.

Continuous delivery and continuous deployment are illustrated in Figure 2.2.

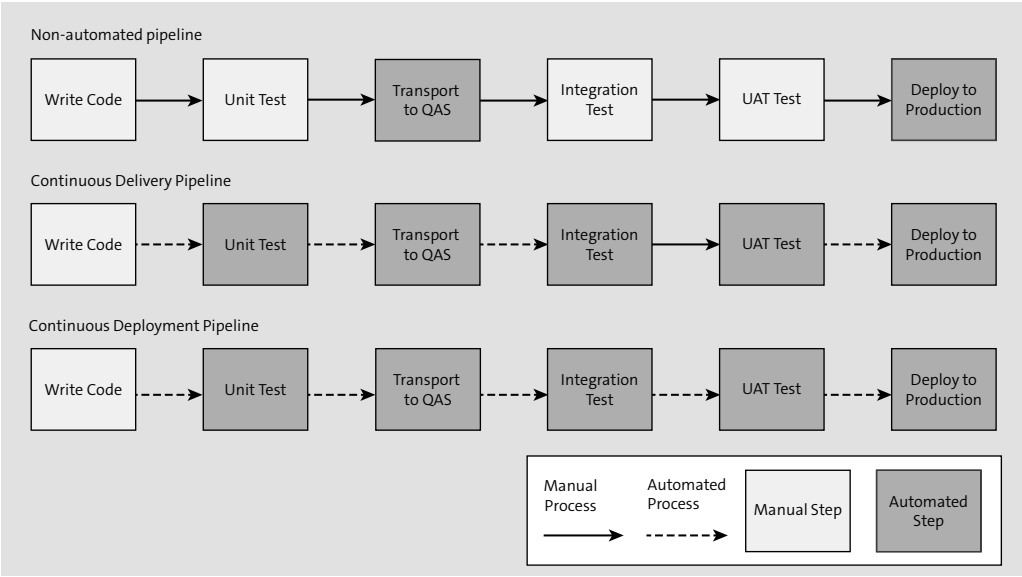


Figure 2.2 DevOps: Continuous Delivery and Continuous Deployment

To support DevOps methodology for software development, SAP HANA XS Advanced supports external source code management tools, such as Git or Gerrit, to be used as a central repository (discussed in Section 2.6) while streamlining the build and deployment process using multi-target application (MTA) archives (.mtar) and all-or-nothing deployment.

With SAP HANA XS Advanced, the system landscape can be a simple two-tier landscape (DEV-PRD) with the QA layer provisioned in the DEV landscape or a standard three-tier landscape (DEV-QA-PRD), as illustrated in Figure 2.3. It's also possible to provision additional systems (QA, SIT [system integration testing], UAT, etc.) on demand to support further testing for specific applications, creating a complex system landscape (DEV-QA-SIT-UAT-PRD).

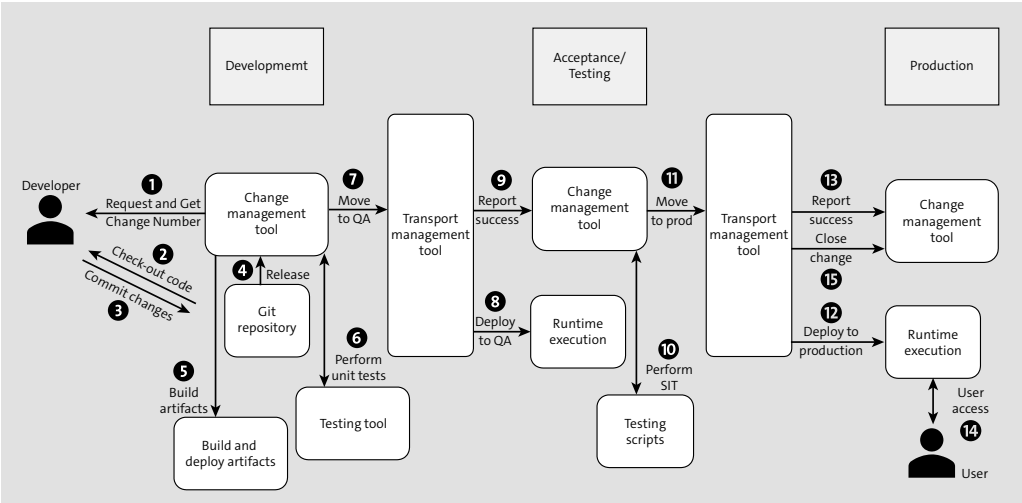


Figure 2.3 Application-Specific Landscape

Because the source code is centrally stored and managed in an external dedicated repository such as Git or Gerrit, the DEV environment can be provisioned on premise or in the cloud. In addition, it can be a shared system within the organization or can be specific to a team or developer. The application can be built based on the committed changes to the central repository, which then can be deployed to any system in the landscape.

Now that you have a fair understanding of the system landscape, let's discuss your options for provisioning a development system.

2.2 SAP HANA, Express Edition

SAP HANA, express edition was officially introduced during SAP TechEd in Las Vegas in September 2016. It's a streamlined version of SAP HANA, optimized to run in computers with minimal hardware configurations, such as personal laptops and desktop systems. It can also be installed and provisioned in public cloud platforms, such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, and Docker Store.

SAP HANA, express edition comes as a binary installation package for Linux Intel systems and/or as a virtual machine (VM) for any operating systems capable of running a hypervisor, and it can be downloaded free of cost from the SAP website. Developers, startups, and companies can use SAP HANA, express edition to develop, deploy, and run applications in production free of cost up to 32 GB, which can be scaled up to 128 GB for an additional fee.

To efficiently run in a smaller hardware configuration and to support application development in PCs, the following features of the SAP HANA platform have been excluded from SAP HANA, express edition:

- Data warehousing foundation
- Disaster recovery
- Dynamic tiering
- High availability
- Multihosting
- Outward scaling for multiple hosts
- Remote data synchronization
- SAP Solution Manager
- SAP HANA smart data integration (SDI)
- SAP HANA smart data quality (SDQ)
- SAP HANA streaming analytics
- System replication

SAP HANA, express edition can be installed on premise or in the cloud, and the installation and configuration steps are discussed in the following sections.

2.2.1 On-Premise Installation Options

As stated earlier, SAP HANA, express edition is available as a binary installation for Linux and as a VM image for any other operating systems capable of running a hypervisor, as follows:

■ Binary installation for Linux

Binary installation packages are available for both SUSE Linux Enterprise Server 11.x or 12 and Red Hat Enterprise Linux 6.x.

■ VM image

This VM package offers the simplest setup and is platform-independent so it can be used in Linux, Microsoft Windows, Mac OS, or any other operating system capable of running a hypervisor and with at least 16 GB of RAM. The following hypervisor are supported:

- VMware Player 7.1
- VMware Workstation Pro 12.1
- VMWare Fusion 8.x
- VMWare Fusion Pro 8.x
- Oracle VM Virtual Box 5.0.14 or higher

Both the binary installation and VM image options come in two flavors: *server-only package* and *full version package*. The latter includes additional software components, such as SAP HANA cockpit, SAP HANA XS Advanced services, and SAP Web IDE for SAP HANA. We recommend installing the full version package or use the full version VM image to work with the examples discussed in this book. The binary installer and/or the VM image can be downloaded from the SAP HANA, express edition website at www.sap.com/sap-hana-express.

2.2.2 Installing SAP HANA, Express Edition VM Images

In this section, we'll discuss the steps to install SAP HANA, express edition 2.0 SPS 3.0 using the VM images. It will install the following:

- A VM running SUSE Linux Enterprise Server (SLES) for SAP Applications 12 SP 2
- An SAP HANA, express edition 2.0 SPS 2.0 instance on the VM with SAP HANA cockpit, SAP HANA XS Advanced services, and SAP Web IDE for SAP HANA preconfigured

Before we proceed with the installation, ensure the following prerequisites are met by the computer to install SAP HANA, express edition:

- At least 8 GB of RAM is required for the *server-only* installation. At least 16 GB of RAM is required (24 GB RAM is recommended) for the *full package* (server plus applications).
- At least two CPU cores (four CPU cores are recommended) are required with virtualization support. For Intel processors, virtualization is a BIOS setting known as either *Intel Virtualization Technology* or *Intel VT*.
- At least 120 GB of storage (Solid State Disk [SSD] recommended) is required.
- A 64-bit Java SE Runtime Enterprise 8 (JRE 8) is installed and running in the machine. JRE 8 can be downloaded from www.java.com.

Follow these steps to proceed with the installation:

1. Install the hypervisor.

Hypervisors are software applications used for creating and running VMs. We'll be using VMware Player in our example, which can be downloaded from www.vmware.com.

2. Register.

Go to the SAP HANA, express edition web page at www.sap.com/sap-hana-express, and register to download the software package. Upon successful registration, select the OS-specific (Windows, Linux, or platform-independent) download manager link under **1A. ON-PREMISE INSTALLATION**. Save the download manager executable.

3. Download the software using Download Manager.

Execute the Download Manager in GUI mode. Select **Linux/x86-64** for **Platform**, **Virtual Machine** for **Image**, and the appropriate **Save** directory. Select **Server + applications virtual machine** option (**Getting Started with SAP HANA, express edition (Virtual Machine Method)** is selected by default), and click the **Download** button, as illustrated in Figure 2.4. A popup message will confirm the successful download.

Note

Additional packages can also be selected, but it's efficient to use the VM's built-in Download Manager.

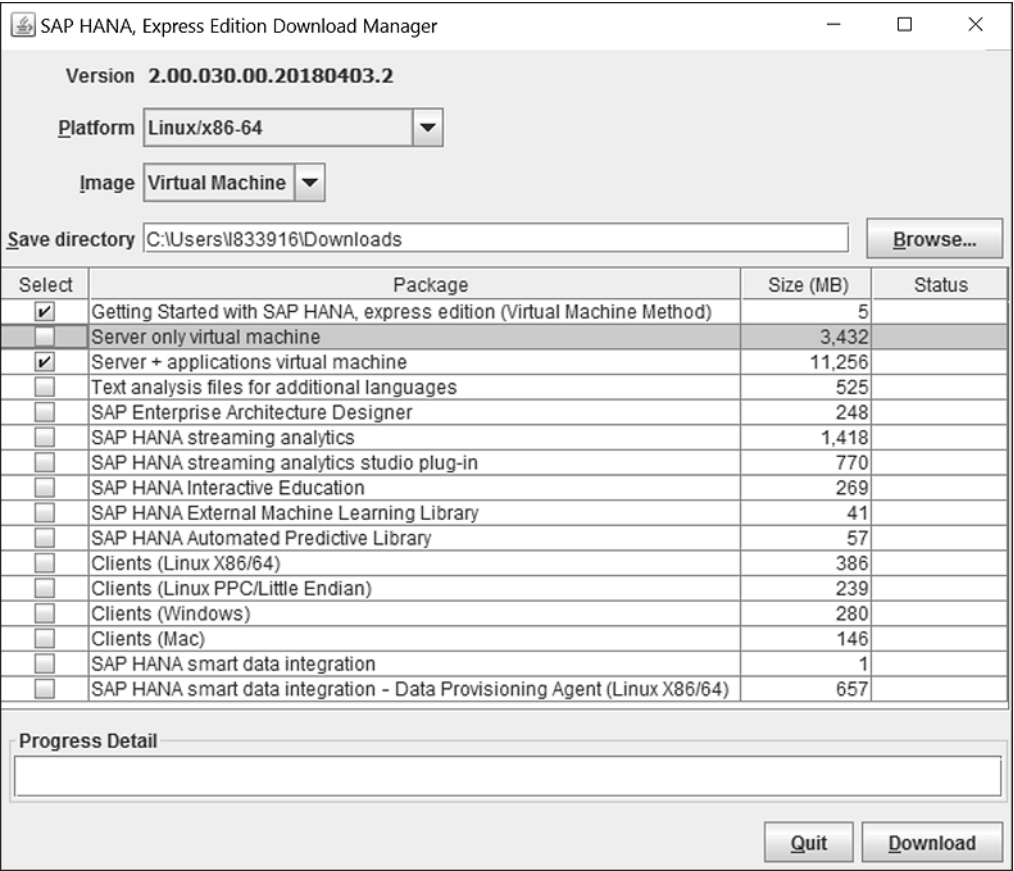


Figure 2.4 SAP HANA, Express Edition Download Manager

4. Import the open virtual appliance (OVA) file.

Start the VMware Player, and select **Open a Virtual Machine** to open the downloaded OVA *hxexsa.ova* file (for **Server + applications virtual machine**). Accept the default options, and import the VM. It will take approximately 5–12 minutes to complete the process.

5. Turn on the VM and connect.

Power on the VM using the **Play** button as shown in Figure 2.5. The IP address of the VM is displayed in the login screen. You’ll need the IP address to configure and connect to your SAP HANA, express edition system. If the IP address isn’t shown, power off the VM, and restart it again. Use **Host Name: IP address** and **port: 22** to connect to the VM instance.

Note

If you’re using an English QWERTY keyboard, you can continue using the default VM console. However, if you’re using another keyboard (German QWERTZ or French AZERTY), you must use a Secure Shell (SSH) client, such as PuTTY. PuTTY can be downloaded free of cost from www.putty.org.

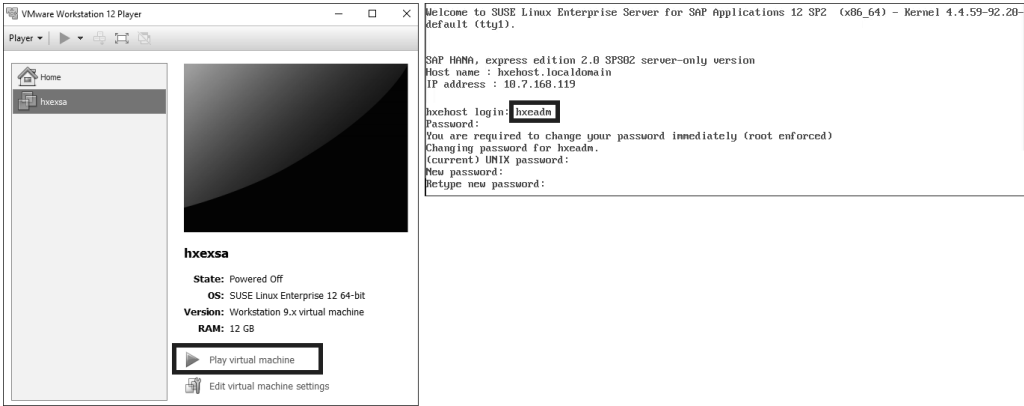


Figure 2.5 SAP HANA, Express Edition VM Login and Setup

6. Set the user name and password.

Use the default user name “hxeadm” and temporary password “HXEHana1” (case sensitive). When prompted to change the temporary password, select a new strong password that complies with the following rules:

- At least eight characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- Can contain special characters, but not ` (backtick), \$ (dollar sign), \ (backslash), ‘ (single quotation mark), or “ (double quotation marks)
- Can’t contain dictionary words
- Can’t contain simplistic or systematic values, such as strings in ascending or descending numerical or alphabetical order

If the password isn’t strong enough, the system will log off, and you’ll have to log in again. When prompted, reenter the password to confirm, and change the password Linux OS user password.

7. Set the master password.

The system will then prompt you to change the SAP HANA database master password (i.e., password for the SYSTEM user). In our example, because we’re installing the server + applications VM, it also changes the password for the XSA_ADMIN and XSA_DEV users.

When prompted, select a strong password (it can also be the same as the OS user password), and reenter the password to confirm and change it.

8. Allow or disallow Internet access.

When prompted with **Do you need to use the proxy server to access the internet?**, select **Y** or **N** as appropriate. If you’re inside a corporate firewall, you may have to select **Y** and enter the proxy host name, port number, and a comma-separated list of hosts that don’t need a proxy, which must include localhost, hxehost, and hxehost.localdomain.

9. Complete the installation.

Select **Y** for the **Wait for the XSA configurations to finish?** to complete the SAP HANA XS Advanced configuration before starting the server.

Finally, the summary of the configuration before execution will be displayed. Select **Y** when prompted to **Proceed with configuration?**. SAP HANA cockpit and SAP HANA XS Advanced configurations will take around 10 minutes. If the VM goes black, click **VM** or press the **Ctrl** key to wake it up.

When the installation and configuration is completed, the **Congratulations! SAP HANA, express edition 2.0 is configured** message will be displayed.

Now that the installation is complete, there are some post-installation checks to be done. You need the SAP HANA server host (hxehost) IP to connect to the SAP HANA database using client tools. This is the same IP address that was used to connect using the SSH/PuTTY client, and it can be displayed by using the following code in the VM command prompt:

```
/sbin/ifconfig code
```

The hxehost IP address is local/private to the VM. To enable applications (e.g., a web browser or SAP HANA Studio) to access the hxehost from your laptop, the hxehost IP address should be added to the laptop’s host file (e.g., *c:\Windows\System32\Drivers\etc\hosts* for Windows and */etc/hosts* for Mac).

For Windows, create an entry with the following format in the *hosts* file and save it:

```
<hxehost IP address> hxehost
```

For Mac OS or Linux, use the following code in the command prompt:

```
sudo sh - c 'echo <hxehost IP address> hxehost >> /etc/hosts'
```

The system database of the just installed SAP HANA, express edition instance can be accessed using the following connection parameters in SAP HANA Studio:

- **Hostname:** hxehost (or hxehost IP address)
- **Instance Number:** 00
- **Mode:** Multiple Container and System Database

You can also check the status of the SAP HANA extended application services, classic model (SAP HANA XS) server by using the following URL in the browser: *http://hxehost:8090*.

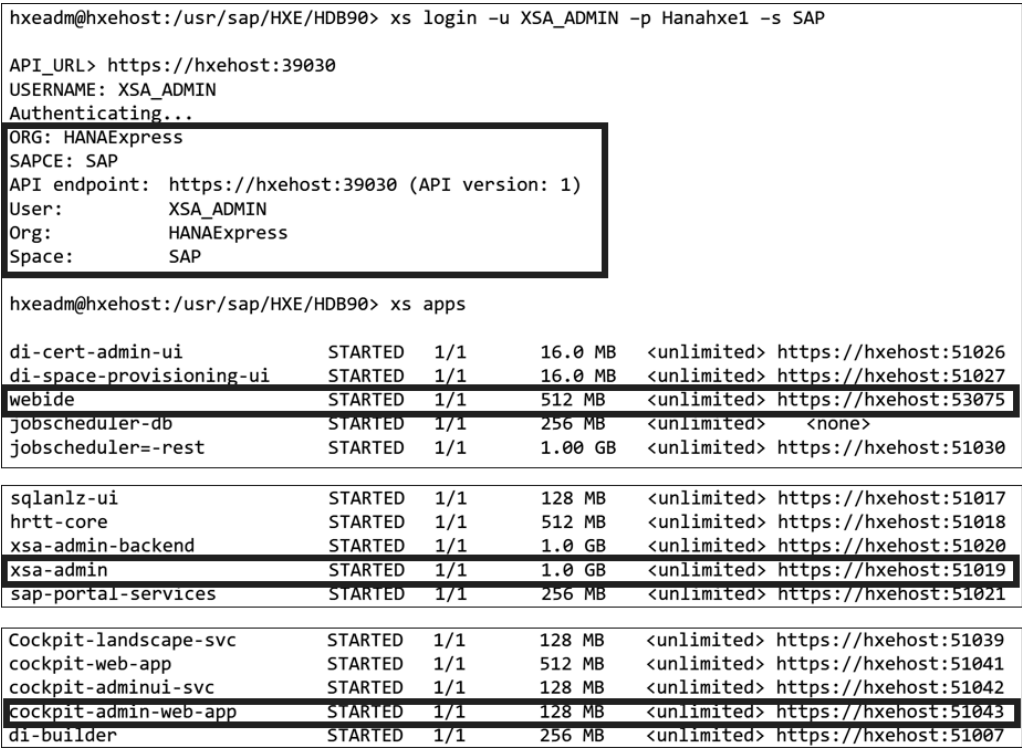


Figure 2.6 Check SAP HANA XS Advanced Apps

To check the status of SAP HANA XS Advanced applications in the system, use the VM command prompt to connect to the SAP HANA XS Advanced server, as shown in Figure 2.6. We'll use the following commands to connect to the SAP HANA XS Advanced server:

```
xs login -u XSA_ADMIN -p "<password>" -s SAP
```

In this code, <password> is the password used to install the SAP HANA database. This will connect to the SAP HANA XS Advanced services and display the following information:

- **ORG:** HANAExpress
- **SPACE:** SAP
- **API endpoint:** https://hxehost:39030 (API version: 1)
- **User:** XSA_ADMIN
- **Org:** HANAExpress
- **Space:** SAP

If the **API Endpoint**, **User**, **Org**, and **Space** are displayed, the SAP HANA XS Advanced installation is working as expected.

The SAP HANA cockpit and SAP Web IDE for SAP HANA are also installed as SAP HANA XS Advanced applications. You can check the status of SAP HANA XS Advanced applications using the following code: `xs apps`. This will display all the SAP HANA XS Advanced applications installed and their execution status, as shown previously in Figure 2.6. Look for the `cockpit-admin-web-app`, `XSA-ADMIN`, and `webide` applications, which should show as **STARTED** with 1/1 instance. Record the URLs of those applications to test them in the browser.

2.3 SAP Cloud Appliance Library

The SAP Cloud Appliance Library (<https://cal.sap.com>) provides an online library of preconfigured SAP solutions that can be deployed in the cloud in minutes and can be used for testing, demoing, training purposes. SAP Cloud Appliance Library doesn't host the solution; rather, the solution instance is hosted by the customer's preferred cloud providers, such as AWS and Azure. All deployed solution instances are easily accessed and managed using the SAP Fiori-based self-service portal.

In the following sections, we'll discuss the steps to configure an SAP HANA, express edition system in an AWS cloud using SAP Cloud Appliance Library.

2.3.1 Registrations

You'll need to register and create an account in SAP Cloud Appliance Library to start using the available preconfigured SAP solution. SAP Cloud Appliance Library is free to try for 30 days, but customers will be charged by the cloud providers based on their usage during this trial period. As the solutions are hosted by cloud providers such as AWS and Azure, you should register and create an account with your preferred cloud provider.

The **AWS Access Key** and **Secret Key** or **Azure Subscription ID** is required to connect your cloud providers to the SAP Cloud Appliance Library account, as shown in Figure 2.7.

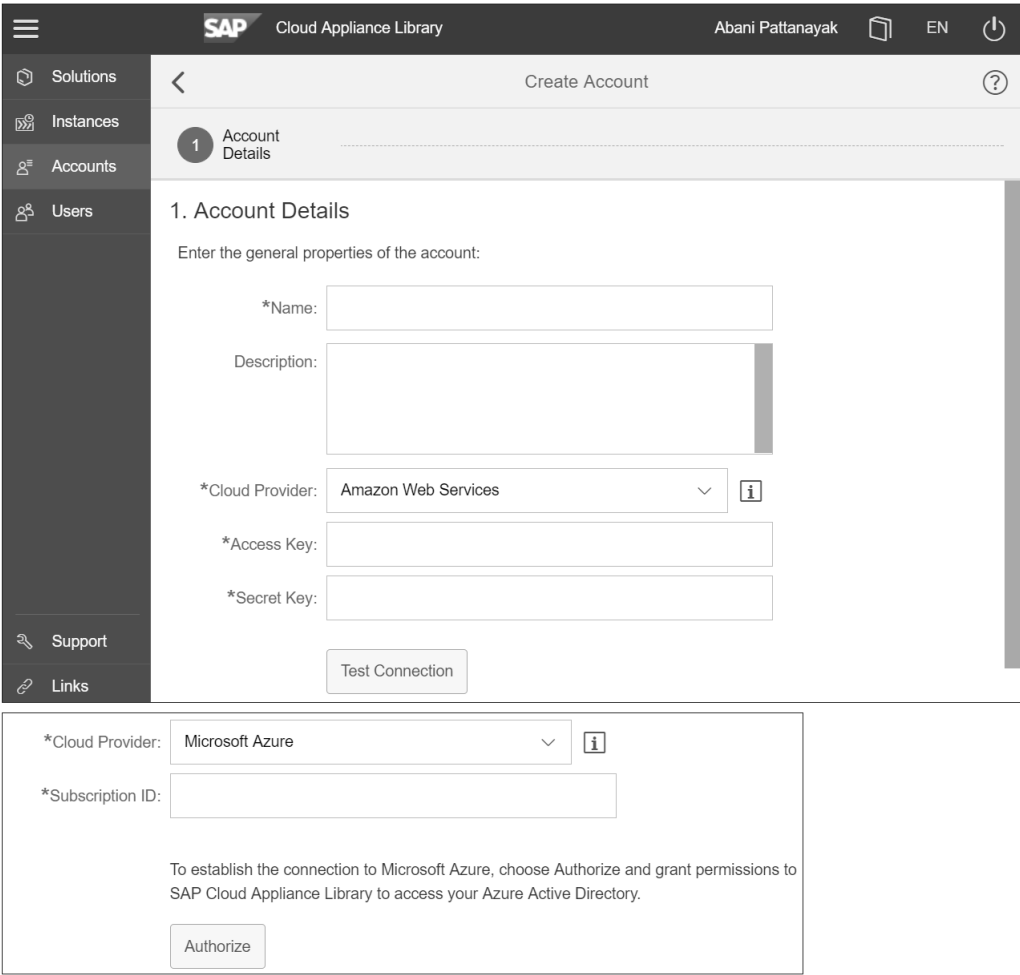


Figure 2.7 Link Cloud Providers

2.3.2 SAP HANA, Express Edition Setup

In the following sections, we'll discuss the steps to set up an SAP HANA, express edition instance in SAP Cloud Appliance Library.

Create Instance

To create an instance of SAP HANA, express edition, navigate to the **Solutions** tab, look for **SAP HANA, Express Edition** and select the **Create Instance** button, as shown in Figure 2.8 ❶.

Select the **Advanced Mode** button to proceed to the next screen to select account details, as illustrated in Figure 2.8 ❷ and ❸.

Select **SAP Cal Default Network** (for Azure) and **ECS-Internet** (for AWS) in the **Network** field to have an external IP to connect to the instance, as illustrated in Figure 2.8 ❹.

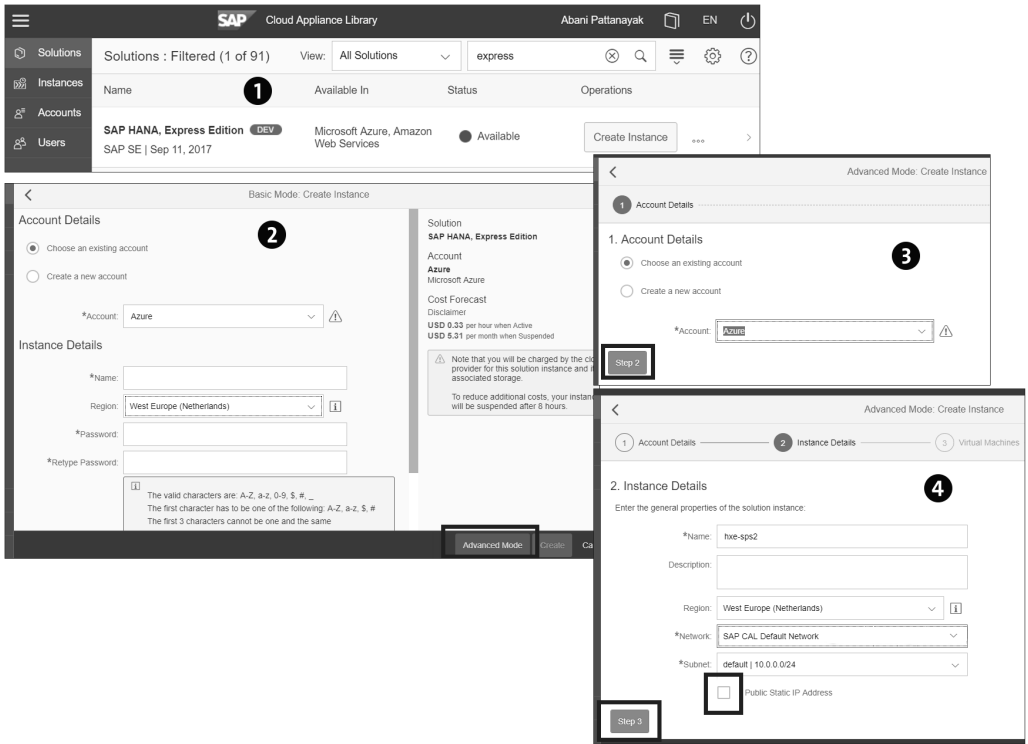


Figure 2.8 Creating an Instance

Review and record the access ports shown in Figure 2.9 ❶. Proceed to the next step to select the SAP HANA master password (for SYSTEM user) and XSA_ADMIN and XSA_DEV user. Make sure the password follows the strong password rules shown in Figure 2.9 ❷.

Proceed to the next screen to select the appropriate scheduling options, as illustrated in Figure 2.9 ❸. Go to the next screen to review the instance details, and click **Create** to create the instance, as shown in Figure 2.9 ❹.

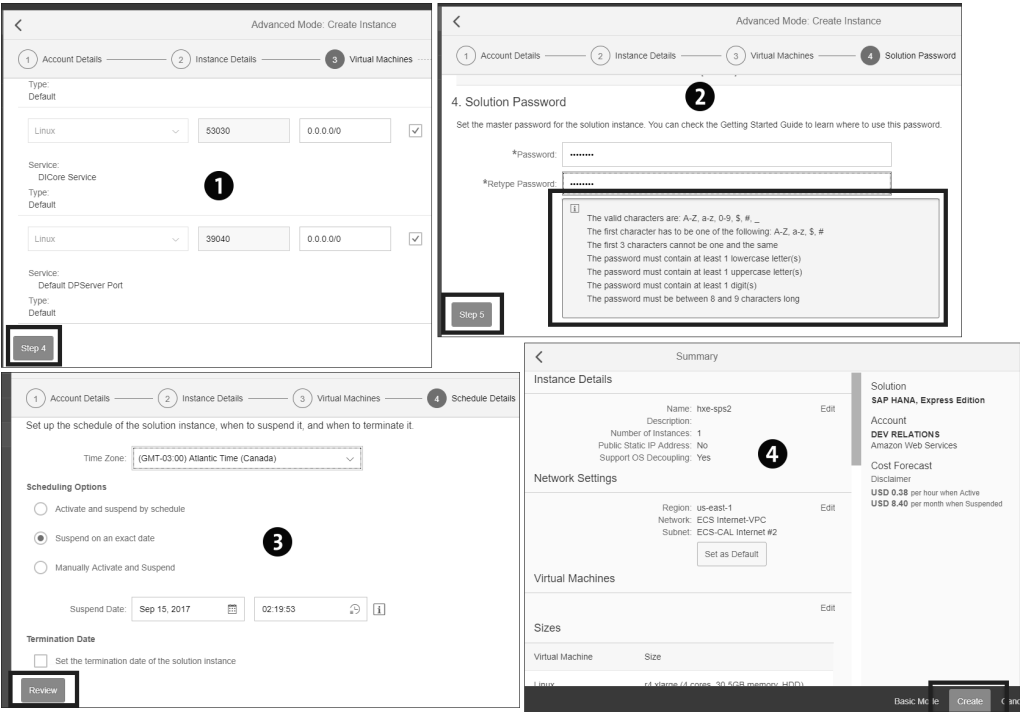


Figure 2.9 Instance Details

The wizard will then display a disclaimer about the usage of SAP HANA, express edition. Upon confirmation, you'll be prompted to store and download the private key. Select **Store** to save the private key details to your instance, and then click **Download** to download it to your local system to connect to the SAP HANA instance using an SSH client such as PuTTY.

It will take around 20–25 minutes to provision and configure the SAP HANA, express edition system in the cloud.

Post-Configuration Checks

When the instance is available and ready for use, it will be displayed in the **Instances** tab. Click on the instance to access its details, record the external IP address, and download the private key, which can be used to connect to the SAP HANA instance from your local computer, as illustrated in Figure 2.10.

Download the *Getting Started Guide* to find details about connecting your SAP HANA, express edition instance, including the SAP HANA database server details (server ID [SID], instance number, database user names, OS user names) and port details of back-end services (e.g., port 3xx30 for the SAP HANA XS Advanced server and port 5xx75 for SAP Web IDE for SAP HANA, where xx is the instance number).

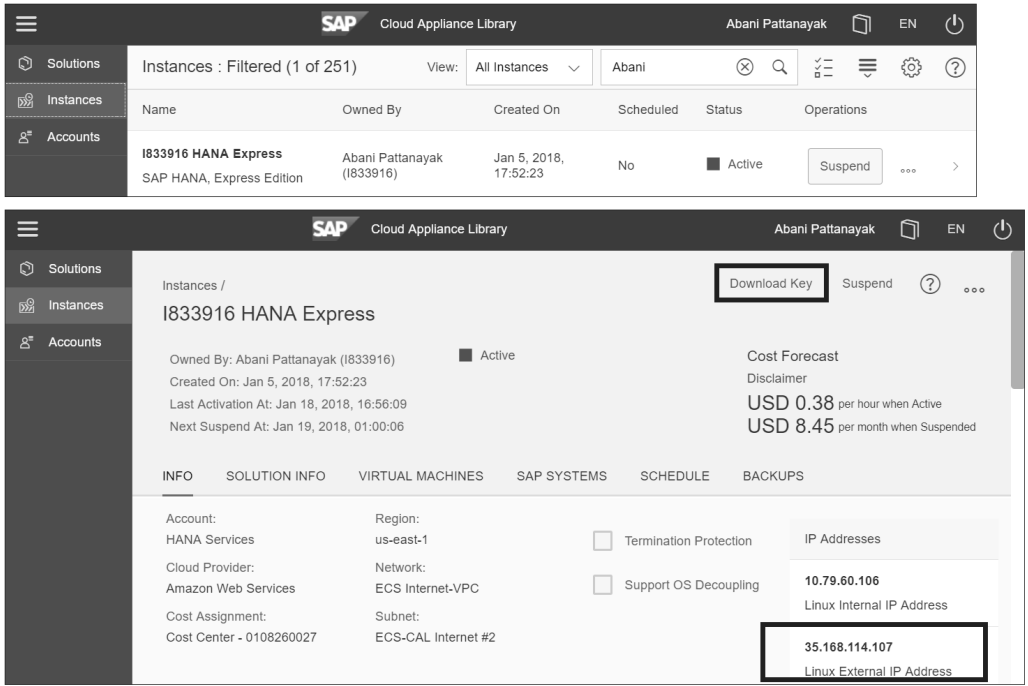


Figure 2.10 SAP CAL Instance Details

To seamlessly access all SAP HANA XS Advanced applications, the `vhcalhxdb` endpoint must be defined in the local systems by adding the external IP address in the laptop’s host file (e.g., `c:\Windows\System32\Drivers\etc\hosts` for Windows and `/etc/hosts` for Mac OS).

For Windows, create the following entry in the `hosts` file and save it:

```
<IP address>          vhcalhxdb
```

For Mac OS or Linux, use the following code in the command prompt:

```
sudo sh - c 'echo <IP address>          vhcalhxdb >> /etc/hosts'
```

We can then connect to `vhcalhxdb` using SAP HANA Studio, an SSH client (e.g., PuTTY), and a browser to SAP HANA XS Advanced web applications.

Accessing SAP HANA, Express Edition

The downloaded private key file should be converted to a `.ppk` file using key generator tools such as PuTTYgen. Use the generated `.ppk` private key file to authenticate and default the `root` user to connect to the Linux host of the SAP HANA database using SSH tools (e.g., PuTTY).

Use the following commands in the SSH prompt to check the status of the SAP HANA instance. `HXADM` is the Linux OS user in SAP HANA:

```
su - hxeadm
HDB info
```

If SAP HANA is up, the `SYSTEM` database of the just installed SAP HANA, express edition instance can be accessed using the following connection parameters in SAP HANA Studio:

- **Hostname:** `vhcalhxdb`
- **Instance Number:** `90`
- **Mode:** **Multiple Container and System Database**

The SAP HANA XS Advanced server (and web dispatcher) status can be checked using the following: `https://vhcalhxdb:3xx30` (where xx is the instance number). This will list all available SAP HANA XS Advanced applications and their URLs. For example, the SAP HANA XS Advanced administration console can be accessed using the following: `https://vhcalhxdb:51015`. Similarly, SAP Web IDE for SAP HANA can be accessed using `https://vhcalhxdb:53075`.

The SAP HANA XS Advanced client can be used to connect to the SAP HANA XS Advanced server and check the status of SAP HANA XS Advanced applications such as SAP Web IDE for SAP HANA.

2.4 SAP Cloud Platform

SAP Cloud Platform (formerly SAP HANA Cloud Platform) is an open platform-as-a-service (PaaS) offered by SAP to create new applications or extend existing applications in a secure cloud computing environment. It offers the agility and flexibility to quickly build new or extend existing cloud and on-premise apps (both SAP and non-SAP) with your choice of cloud providers (SAP, AWS, Azure, Google Cloud, etc.) and open and common development languages (Java, Python, JavaScript, Node.js, Cloud Foundry, etc.).

SAP Cloud Platform offers a comprehensive list of services in the area of analytics, business services, data and storage, integration, Internet of Things (IoT), machine learning, mobile services, runtime and containers, software-as-a-service (SaaS) extensions, user experience, and security.

SAP Cloud Platform can be accessed at <https://cloudplatform.sap.com>. It offers a *developer edition* subscription free of cost for evaluation and development purposes that includes the following:

- 1 GB shared SAP HANA
- SAP HANA Cloud Portal
- SAP Web IDE
- SAP Cloud Platform mobile services
- SAP Cloud Platform OData Provisioning

Note

The SAP HANA XS Advanced architecture is loosely based on Cloud Foundry and shares some of its core principles (details about the SAP HANA XS Advanced architecture appear in Chapter 4), so it's possible to develop applications using the SAP HANA XS Advanced environment and run/deploy the applications in the SAP Cloud Platform Cloud Foundry environment.

For SAP HANA XS Advanced application development, subscriptions to the following two services are required:

- SAP HANA service instance
- SAP Web IDE for full-stack development

As of writing this book, the SAP HANA service instance isn't offered as part of the developer edition or trial subscription because it requires at least 16 GB of memory.

In the following sections, we'll discuss provisioning the SAP HANA service instance and SAP Web IDE for full-stack development in SAP Cloud Platform, which can be used as our development environment.

2.4.1 Registrations

You'll need to register and create an account by visiting the SAP Cloud Platform website at <https://cloudplatform.sap.com> and then purchase a subscription for SAP Cloud Platform, starter edition (32 GB or 64 GB) for nonproductive use or any other subscription for productive use.

SAP Cloud Platform offers two environments: Cloud Foundry (infrastructure by AWS, Google Cloud Platform, and Azure) and Neo (infrastructure by SAP). We'll be using the *Cloud Foundry* environment for the SAP HANA service and the Neo environment for the SAP Web IDE, which is discussed in the following sections.

2.4.2 Provisioning the SAP HANA Service Instance

In this section, we'll discuss the steps to provision an SAP HANA service instance in the Cloud Foundry environment of SAP Cloud Platform. The SAP HANA service is fully managed by SAP and offers flexible sizing, elastic scaling, and consumption-based pricing and terms of hosting. At the time of writing this book, it's offered in the cloud AWS or Google Cloud Platform. For more information about features and capabilities of the SAP HANA service instance, refer to online help for SAP HANA service.

Set Up the Cloud Foundry Environment

The SAP HANA service is offered in the Cloud Foundry environment of SAP Cloud Platform. The Cloud Foundry environment needs to be set up before you can instantiate any service.

Log in to the **SAP Cloud Platform Cockpit** screen, and navigate to your Cloud Foundry global account. Create one or more subaccounts to organize resources in your global account. Cloud Foundry needs to be enabled explicitly for each subaccount using the **Enable Cloud Foundry** button in the **Overview** page of the subaccount. In our example, we've created a subaccount called **CF-trial**, details of which are illustrated in Figure 2.11 ❶.

Navigate to the **Spaces** page of your subaccount. Create one or more spaces to organize service instances and applications in your subaccount. In our example, we've

created space CF-trial_space, details of which are illustrated in Figure 2.11 ❷. Selecting the space changes the navigation menu on the left side. Navigate to the **Services • Service Marketplace** to display a list of services available (see Figure 2.11 ❸).

Click on **SAP HANA Service** (technical name hana-db) to display the **Overview** page with additional details and a link to documentation of the service. Navigate to the **Instances** page to display the list of instances provisioned for the service as shown in Figure 2.11 ❹.

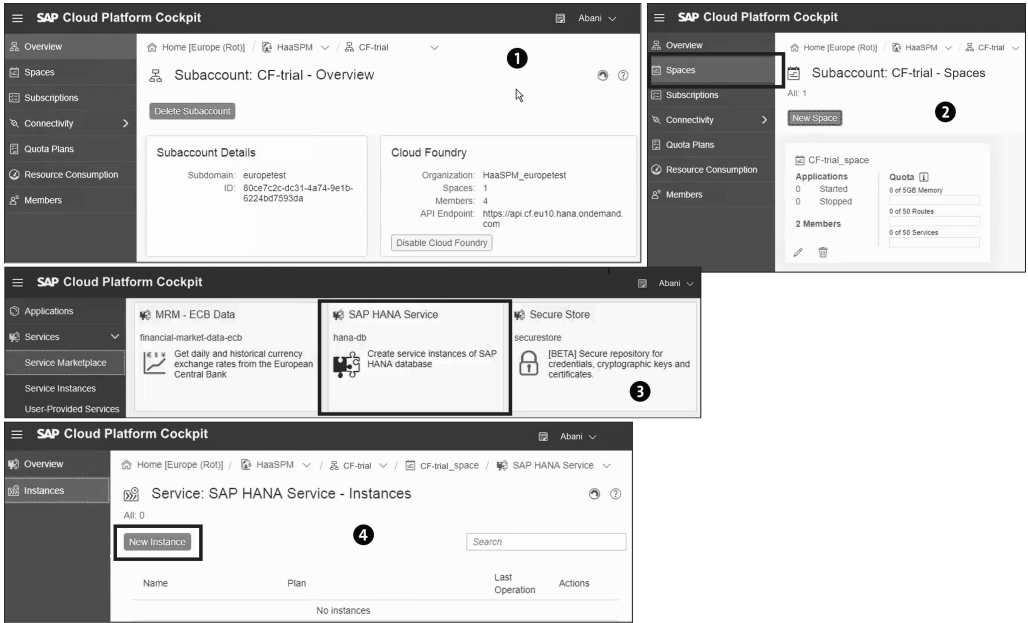


Figure 2.11 Cloud Foundry Environment in SAP Cloud Platform

Create an SAP HANA Service Instance

Click the **New Instance** button to display the new instance wizard for the SAP HANA service as shown in Figure 2.12. There are two plans available for SAP HANA Service: **Standard** (core database features) and **Enterprise** (standard plus additional features, such as predictive, graph, spatial, etc.), as shown in (Figure 2.12 ❶). Select the appropriate service plan based on your business requirement, and click **Next** to proceed.

Specify a password, which will be the password for the system user of the *tenant* database being provisioned. It will actually provision a *system* database and a *tenant* database. As this is an SAP-managed service, the system database is accessed by SAP,

whereas the tenant database is secured using the password and is in your control. Select the appropriate memory (in blocks of 16 GB) to provision the tenant database. Because the SAP HANA instance is going to be available across the public Internet, you can effectively control access to your tenant by whitelisting specific IP addresses or allowing open access. Select the appropriate user parameters, and click **Next** to proceed (Figure 2.12 ❷).

Select **Next** to skip this page because we aren't assigning this tenant database to any particular application (Figure 2.12 ❸).

Specify a name for your instance (e.g., "hana_enterprise" for the enterprise version), and click **Finish** to provision the new instance (Figure 2.12 ❹).

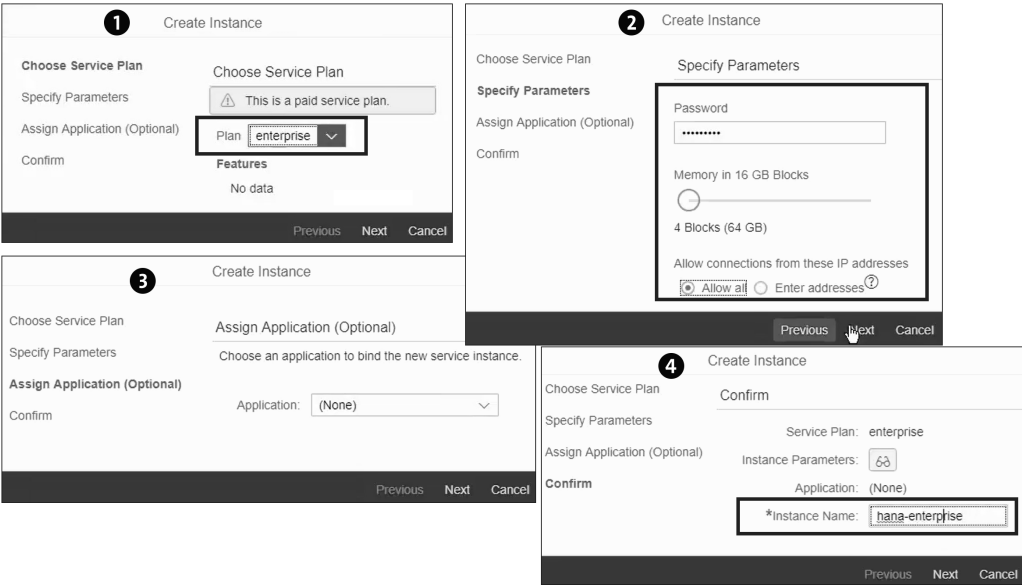



Figure 2.12 Creating an SAP HANA Service Instance

The newly provisioned instance (e.g., hana_enterprise) appears, and it may take a few seconds (up to a few minutes) for the system to be ready for use.

You can spin up as many or as few instances as you want and whenever you want using the SAP Cloud Platform cockpit. Because it's a fully managed SAP HANA service, the high availability backups are done by SAP. However, you control what SAP HANA instances you have and when you want to create/delete them to manage the limit in your quota.

Click on the **Open Dashboard**  icon to open the **SAP HANA Service Dashboard** screen, which will display critical information about the SAP HANA service instance as illustrated in Figure 2.13 ❶. Record the **ID** of the SAP HANA service instance and **Endpoints** details of the tenant and system database, which will be used to connect using client tools ❷. You can also see where the SAP HANA service is hosted (in our example, it's hosted with AWS) and how much memory is allocated for the instance.

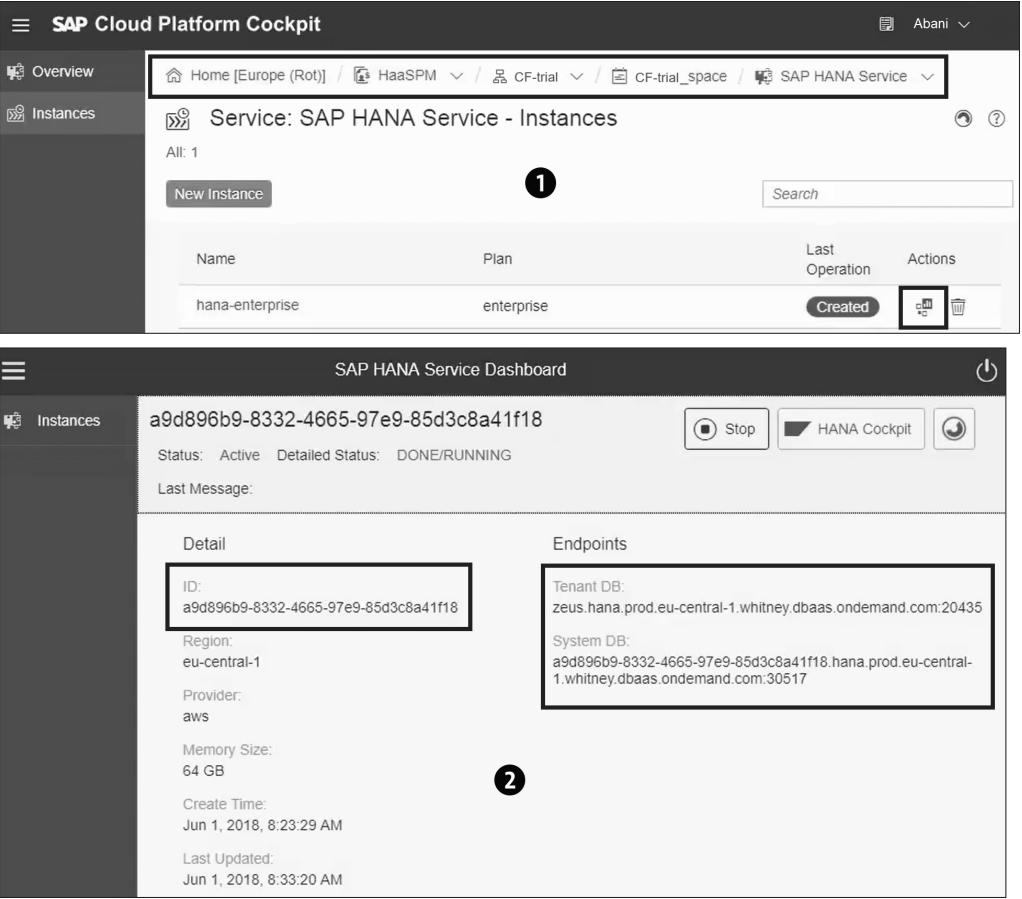


Figure 2.13 SAP HANA Service Dashboard

Accessing the SAP HANA System

Select the **HANA Cockpit** button to open the SAP HANA Platform cockpit of the just-provisioned tenant database in a new browser window. Enter the **SYSTEM** user and the password (used while provisioning the instance) when prompted to authenticate for

the SAP HANA cockpit. The **SAP HANA Cockpit** screen offers various options to effectively manage the tenant database, as illustrated in Figure 2.14 ❶. It's recommended to set up one or more individual users with appropriate roles to perform specific tasks and then deactivate the system user to have a secure environment.

Select the **Open SQL Console** button to launch the **SAP HANA Database Explorer** screen to explore the catalog of schemas and execute SQL commands (see Figure 2.14 ❷).

The provisioned tenant database in the Cloud Foundry environment can also be accessed using SAP HANA Studio with the following connection parameters:

- **Hostname:** <endpoint of the tenant database>, for example, zeus.hana.prod.eu-central-1.whitney.dbaas.ondemand.com
- **Instance Number:** 00 (not required, default 00 is used)
- **Multiple Container & Tenant Database:** H00:<port of the tenant database>, for example, H00:20435

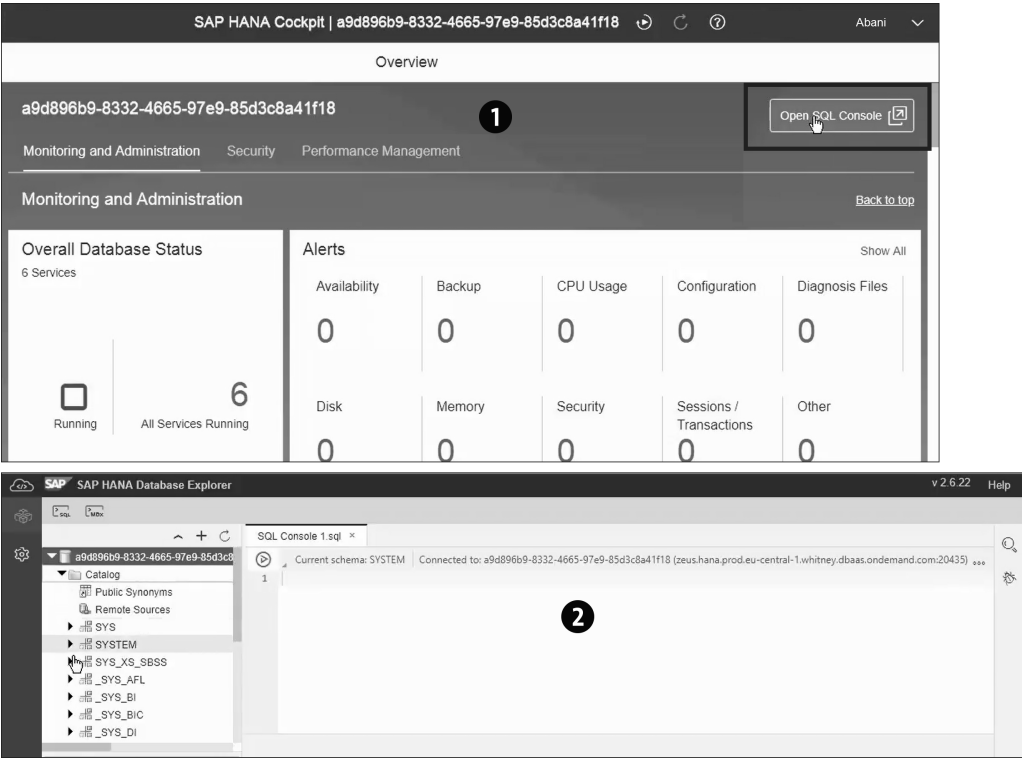


Figure 2.14 Accessing the SAP HANA Service Instance

In the next section, we'll discuss how to enable SAP Web IDE in the SAP Cloud Platform form to work with your SAP HANA service instance.

2.4.3 Enable SAP Web IDE for Full-Stack Development

The SAP Web IDE for full-stack development is used to create multi-target applications (MTAs) and build SAP HANA database artifacts such as database tables, views, and procedures in the SAP Cloud Platform environment. The SAP Web IDE is hosted in the Neo environment of SAP Cloud Platform. To use the SAP Web IDE, you must switch to your Neo subaccount (in our example, it's neo_trial), as shown in Figure 2.15 ❶.

Navigate to **Services** to display the list of services available for your subaccount in the Neo environment, and select the **SAP Web IDE Full-Stack** service to display details about the service (Figure 2.15 ❷). By default, the service isn't enabled. Select the **Enable** button to enable SAP Web IDE for your subaccount (Figure 2.15 ❸).

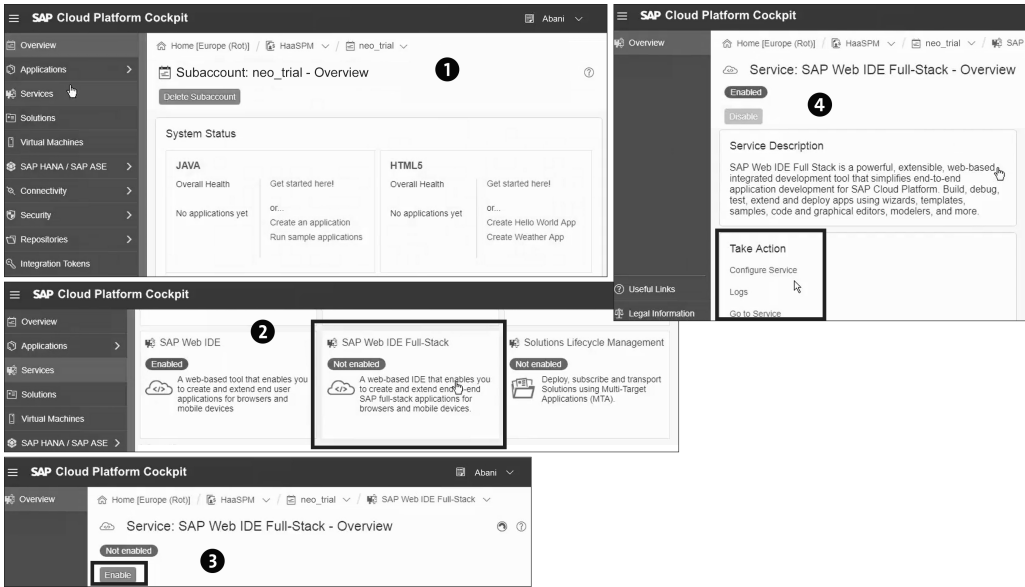


Figure 2.15 Enabling the SAP Web IDE for Full-Stack Development

We need to configure the service and set up users with appropriate roles (e.g., developer and/or administrator, etc.) to make the service ready for users.

Select the **Configure Service** link (Figure 2.15 ❹), select the **DiDeveloper** role, and add one or more users. Similarly, select the **DiAdministrator** role, and add one or more users as illustrated in Figure 2.16 ❶.

We also need to enable principle propagation. To do this, navigate to **Security • Trust**, and explicitly select **Enabled** in the **Principal Propagation** field (Figure 2.16 ❷) instead of using the default setting. Save the changes, and go back to **Services** to launch the SAP Web IDE for full-stack development using the **Go to Service** link (see Figure 2.17 ❶). You can save the URL as a bookmark for easy access.

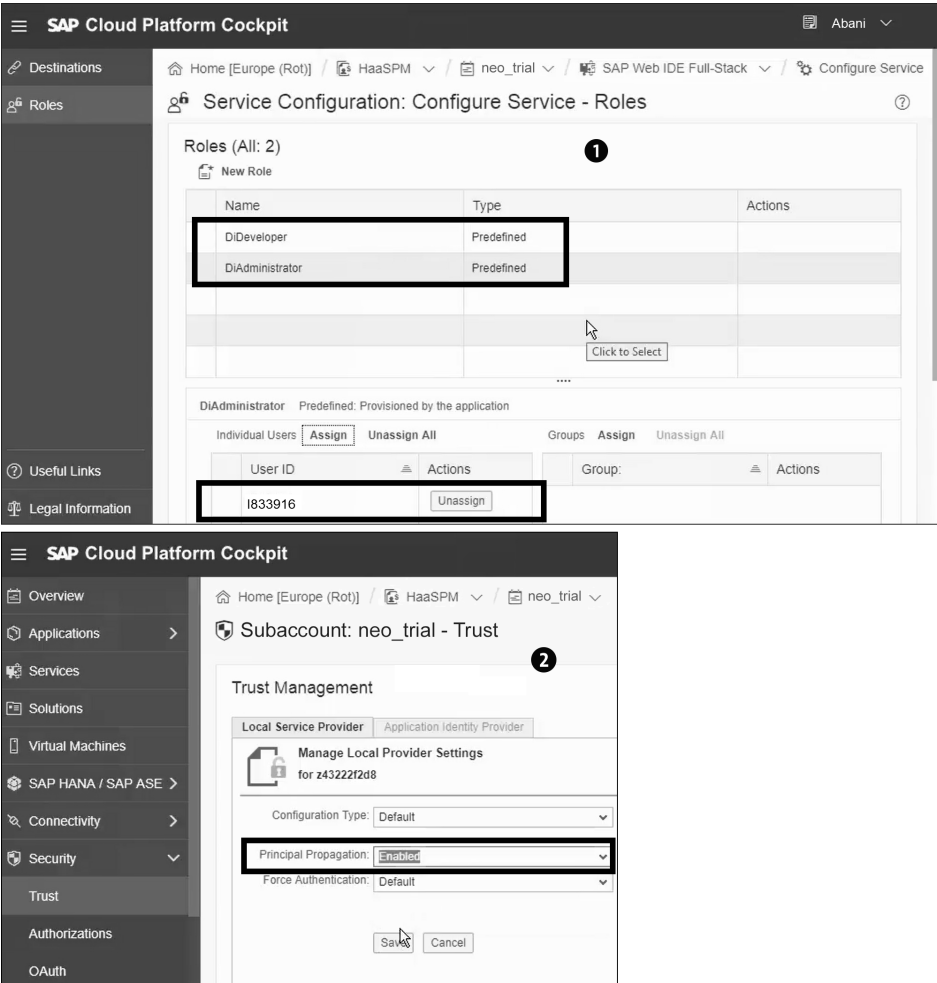



Figure 2.16 Configuring Service Roles

Next, we need to configure the SAP Web IDE to use the SAP HANA service instance we created earlier. In the SAP Web IDE, select the **Preference**  icon, and set up the Cloud Foundry endpoint as illustrated in Figure 2.17 **2**.

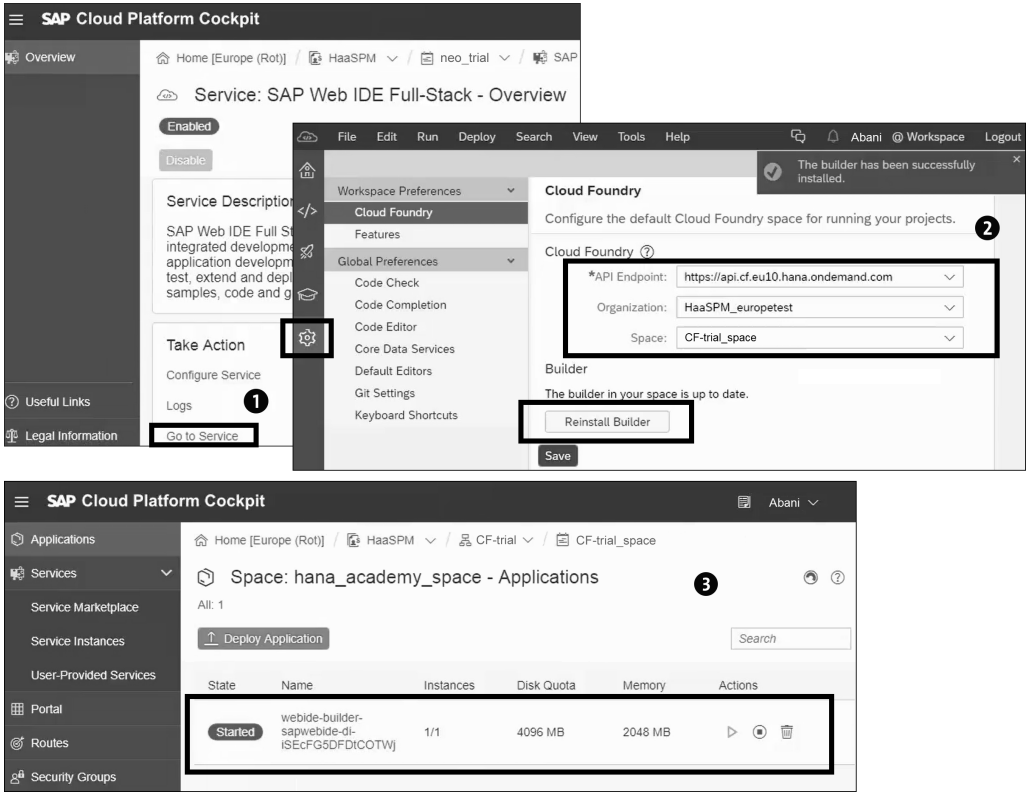


Figure 2.17 Linking SAP Web IDE with the SAP HANA Service Instance

The information for the **Organization** and **API Endpoint** fields is available in the **Overview** page of your Cloud Foundry subaccount as shown earlier in Figure 2.11 **1**, and the **Space** field is the space in which we provisioned the SAP HANA service instance as shown earlier in Figure 2.11 **2**.

Finally, we need to install the builder application that is required to build and deploy applications to our space. To install the builder in your space, select the **Reinstall Builder** button (see Figure 2.17 **2**). Upon successful installation, the builder

application will be available in your Cloud Foundry space under **Applications** as illustrated in Figure 2.17 **3**.

With this, the setup of SAP Web IDE for full-stack development is complete, and it's ready for application development.

2.5 SAP HANA XS Advanced Organizations and Spaces

In the following sections, we'll start by providing you with an overview of the concepts behind organizations and spaces in SAP HANA XS Advanced before moving on to the steps you need to take to manage them.

2.5.1 Organizations and Spaces Overview

Applications in an SAP HANA XS Advanced system are deployed and isolated using the concepts of *organizations* and *spaces* with separated OS users to meet the resource requirements of applications.

An *organization* is a development account that one or more developers (*user*) can own and use. All developers in an organization share a resource quota plan, applications, services availability, and custom domains. All resources, applications, spaces, and services in the organization can be managed (and suspended) at the organization level.

Every application and service is scoped to a space, and each organization contains at least one space. A *space* provides users with access to a shared location for application development, deployment, and maintenance.

If there is a need for a group of applications to share system resources used by the same set of end users and to be deployed by the same user, then this group of applications can be deployed to the same space in an SAP HANA XS Advanced environment. Common resources shared by applications in a space can be data storage, user authorizations, and so on.

Depending on the need for shared resources, one or more applications may be deployed in a given space, but an application is always deployed into one and only one space. Each resource (a service instance) required by an application must be available (or created) in the same space, and each service instance must be explicitly

banded to the application by the service broker. The service binding entity bears the credentials issued by the service broker, and the execution agent passes these credentials to the bound service instances during start-up.

A user represents a developer or collaborator in the context of SAP HANA XS Advanced. Users can have different roles (manager, auditor, and developer) in different spaces within the organization, governing what level and type of access they have within that space. Finally, a space can be used by several users, and each user/developer may have his own private space as well.

One or more spaces can be grouped together as an organization to manage and administrate collectively. However, the runtime behavior of an application isn't impacted by its organization grouping. For all practical purposes, the runtime behavior and the resource requirements are managed at the space level.

The relationships among organizations, spaces, service instances, and service binding are illustrated in Figure 2.18.

The isolation of spaces is achieved by mapping dedicated OS users for each space, as shown in Figure 2.19. Technically, two or more spaces can also be mapped to a single OS user, and those spaces won't be isolated from each other. Applications running in the same space share all resources, such as data storage, user authorizations, and passwords.

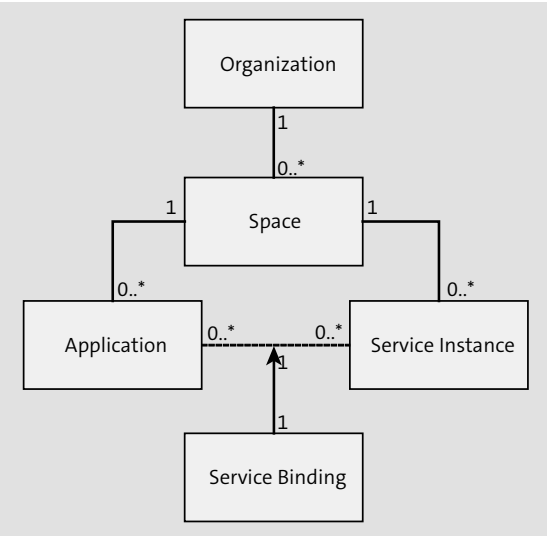


Figure 2.18 Organizations and Spaces

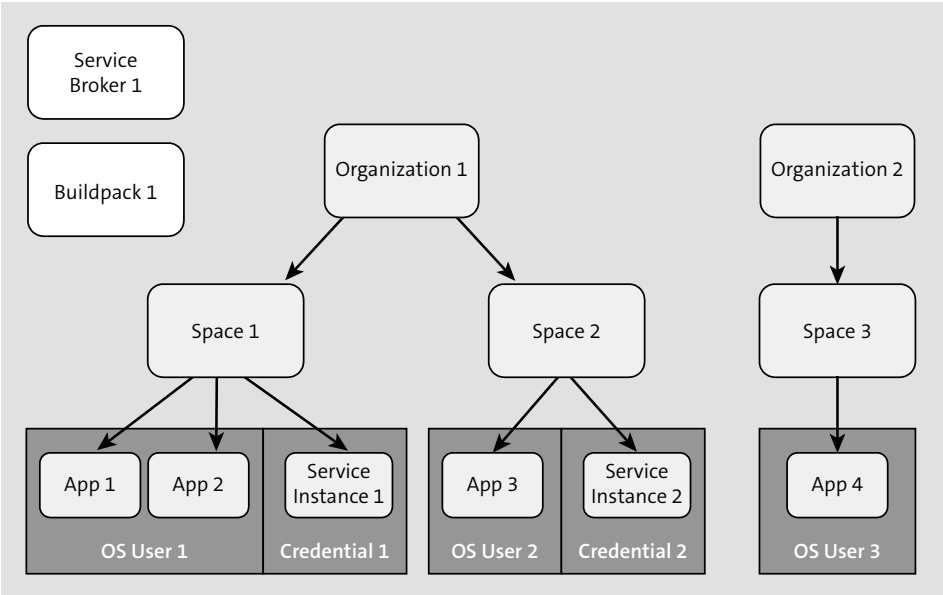


Figure 2.19 OS Level Isolation of Applications

External buildpack processes may or may not have their own dedicated OS user. Application instances and buildpacks in different spaces are only isolated at the OS level if each space is running with a dedicated OS user. From a security perspective, it's important to have isolated spaces with dedicated OS users.

2.5.2 Organization and Space Management

To manage the organization and space for your SAP HANA installation, log in to the SAP HANA XS Advanced administration site. If SAP HANA, express edition is used, SAP HANA XS Advanced administration console can be accessed using <https://hxehost:39030> (on premise) or <https://vhcalhxedb:39030>.

Alternatively, if SAP HANA, express edition isn't used, or the ports have been changed, the right URL for the xsa-cockpit application can be found using command `xs apps` on the command-line interface (CLI). CLI can be accessed directly on the SSH console on the server, or it can be downloaded using the Download Manager. Log in to the CLI using the `xsa_admin` user (or another user with authorizations to create spaces). The CLI is discussed in detail in Chapter 3.

In the following subsections, we'll explain the steps to create a new space and enable it for application development.

Create Space

In the **XS Advanced Administration** console screen, select the **Organization and Space Management** application to display the list of organizations. In our example, the default organization **HANAExpress** is available (Figure 2.20 ❶). The default can be renamed, and a new organization can be created using the **Create Organization (+)** button.

We'll use the default **HANAExpress** organization to create our space. There are two spaces—**SAP** and **development**—already available under the default **HANAExpress** organization (Figure 2.20 ❷). The **SAP** space is used to deploy all SAP-delivered applications that are part of the SAP HANA XS Advanced infrastructure. Select the **+ Create Space** button to create a space, and provide a suitable space name to proceed (see Figure 2.20 ❸ and ❹).

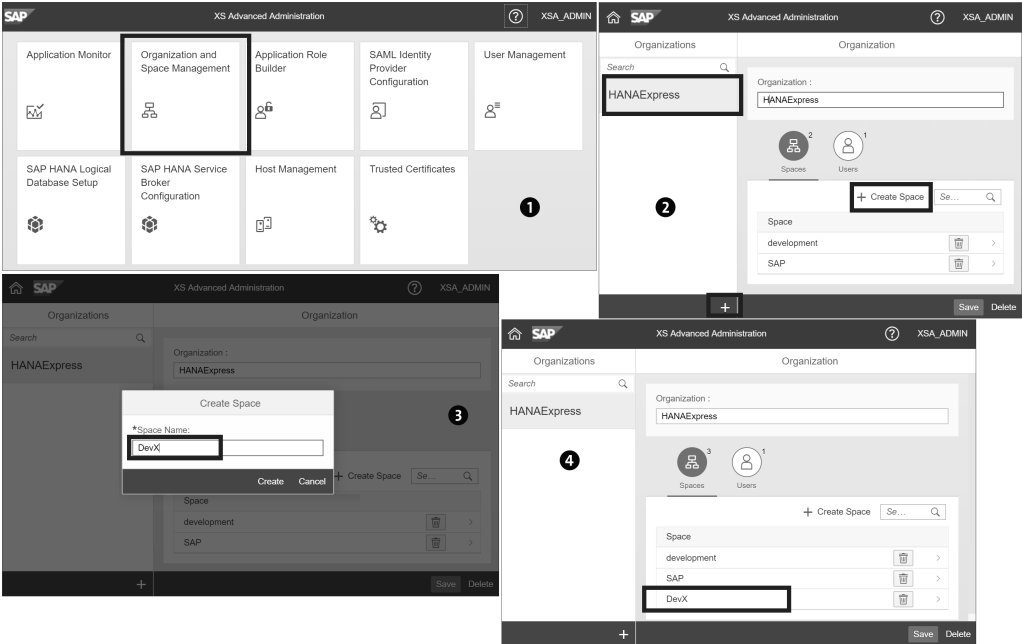


Figure 2.20 Creating a Space

Add Authorized Users

To manage and use the new space just created, users need to be explicitly defined for each space. To do so, click on the new space just created and select the **Users** tab (Figure 2.21 ❶). Select the **Add User** button to select from a list of authorized users and their roles for the space, as shown in Figure 2.21 ❷ and ❸.

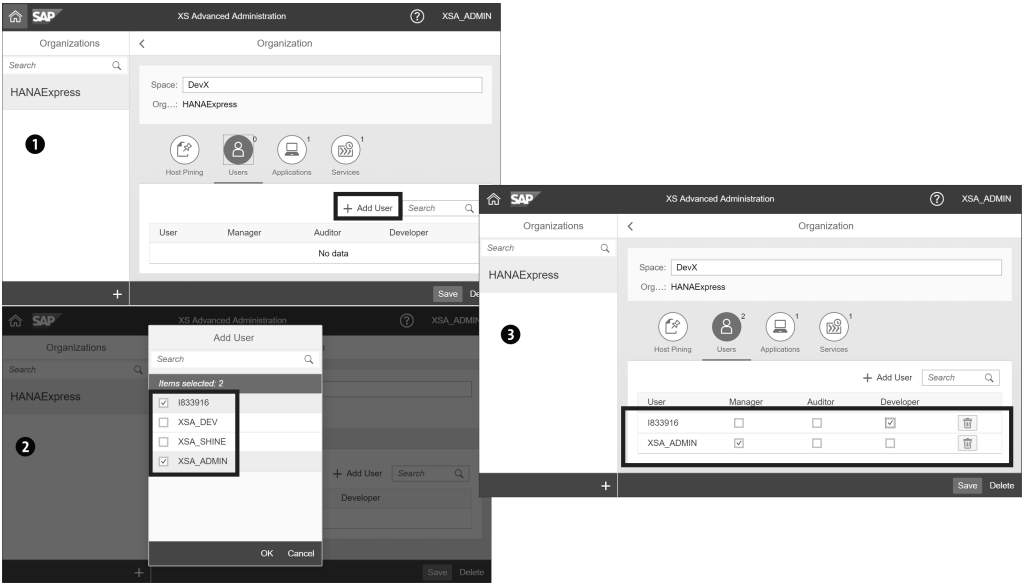


Figure 2.21 Adding Users to a Space

Space Enablement

The new space just created must be explicitly enabled before an SAP HANA XS Advanced application can be deployed in the space. In the **XS Advanced Administration** console screen, select the **Application Monitor** application, and search for app **dis-space-enablement-ui** (Figure 2.22 ❶ and ❷). Start the app, if not running. After the application is running and available, launch the URL (in our example, the app is available on <https://vhcalhxdb:51024>) to open the **Space Enablement** tool.

In the **Space Enablement** tool, select the **Enable** button beside the space (Figure 2.22 ❸). The enablement process may take a few minutes to complete the processing and display a success message ❹. Upon successful enablement, the space is ready for use.

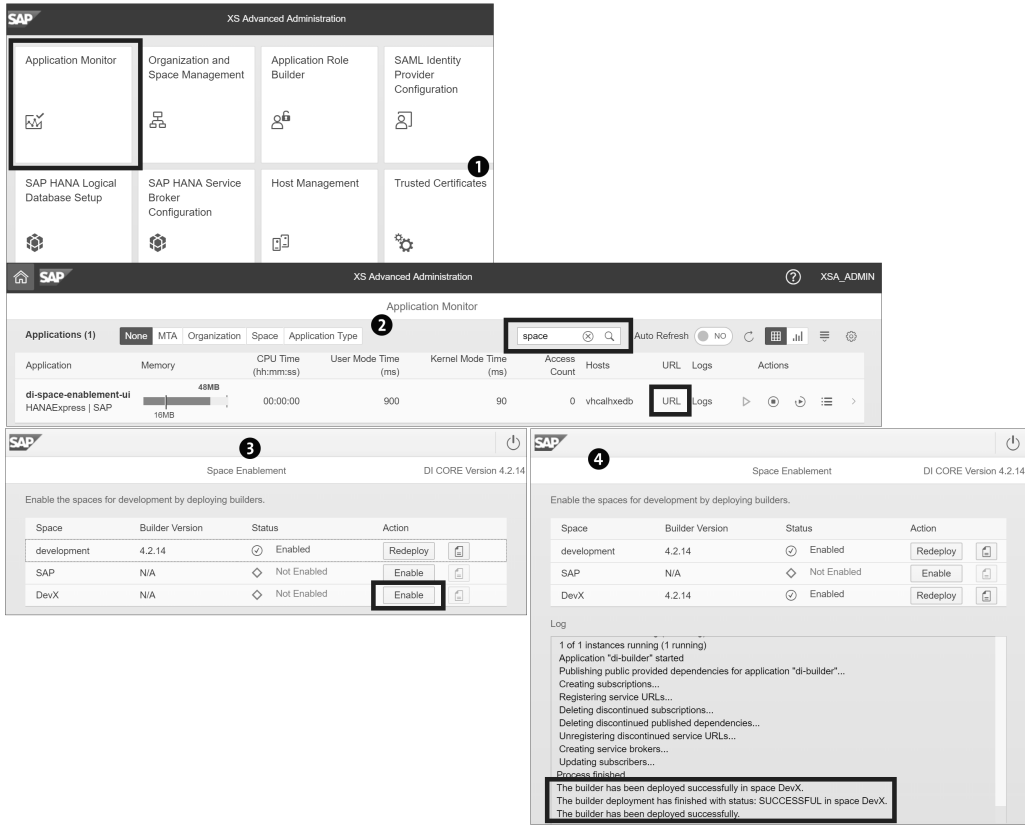


Figure 2.22 Space Enablement

2.6 Git Code Repository

The SAP HANA Repository is available in the SAP HANA system for SAP HANA XS-based development using SAP HANA Studio or the SAP HANA Web-Based Development Workbench. As briefly introduced in Section 2.1, an external source code management tool, such as Git or Gerrit, is used as a central repository for SAP HANA XS Advanced development using SAP Web IDE for SAP HANA.

The Git repository can be hosted in a corporate internal Git/Gerrit installation or a subscription to an external provider such as GitHub, GitLab, or BitBucket. To familiarize

yourself with the Git repository, follow along with our example. First, we'll create our Git repository in GitHub with these steps:

1. Log in to your GitHub account, and select **New Repository** to create a repository.
2. Provide the **Repository Name** and **Description**, check the **Initialize This Repository with a README** option, and click the **Create Repository** button. The repository will be created as illustrated in Figure 2.23.
3. Take note of the Git repository URL, which will be used in setting up the SAP HANA XS Advanced project in SAP Web IDE for SAP HANA.

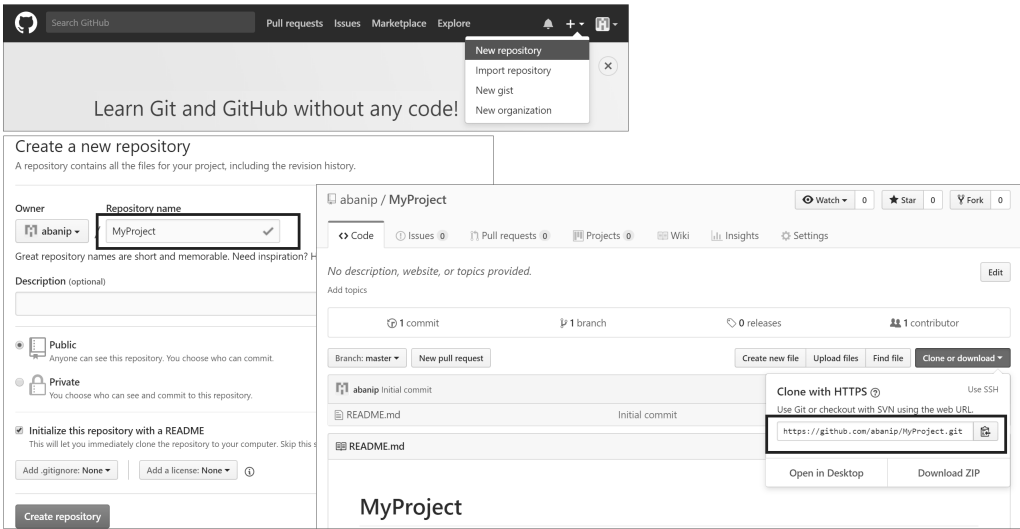


Figure 2.23 Create Git Repository in GitHub

4. Next, begin the project creation by logging in to SAP Web IDE for SAP HANA.
5. Select **File • Project from Template** to select a template for SAP HANA XS Advanced project creation. Choose the **Multi-Target Application Project** template.
6. Provide the **Project Name**, **Application ID**, **Application Version**, **Description**, and **Space** details, and select **Finish** to create the SAP HANA XS Advanced project, as shown in Figure 2.24. The new SAP HANA XS Advanced project will be created and displayed in the SAP Web IDE workspace.

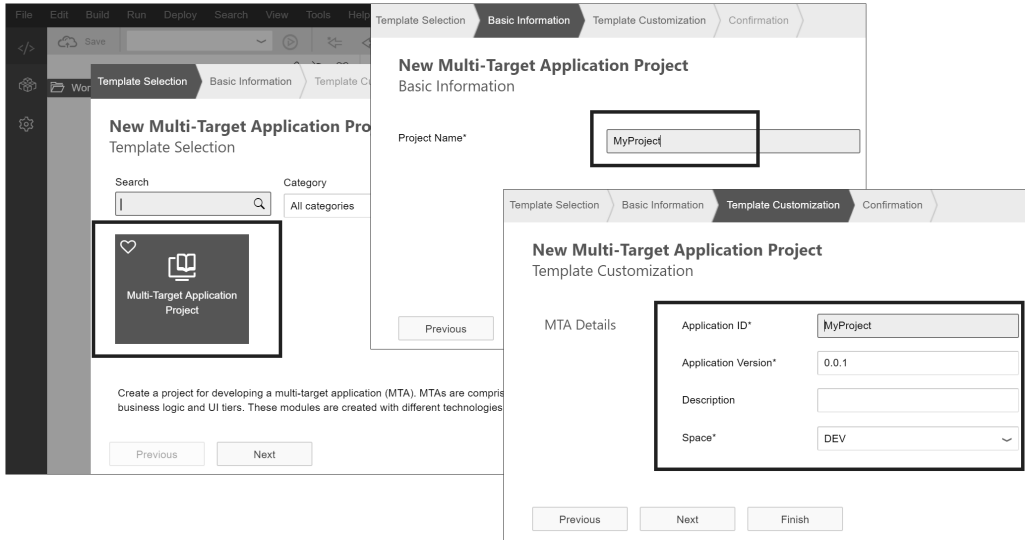


Figure 2.24 Creating an MTA Project

7. To create a local repository for the project, select **Git • Initialize Local Repository** in the context menu shown in Figure 2.25. The successful local repository initialization message will be displayed in the upper-right corner of SAP Web IDE.
8. Select the **Set Remote** option to connect the local repository to the remote Git repository.
9. When prompted, provide the URL of the GitHub repository created earlier. The remote Git server (GitHub in our example) may prompt you to authenticate again to establish the connection.
10. Setting the remote repository automatically does a fetch operation. You should now see the *README.md* file part of the project (the *README.md* file was created during the Git repository creation in GitHub). If the *README.md* file isn't yet pulled (it happens sometimes), select the **Pull** button in the **Git** menu or select **Git • Pull** in the context menu of the project to make sure the local repository has all the files from the remote repository.

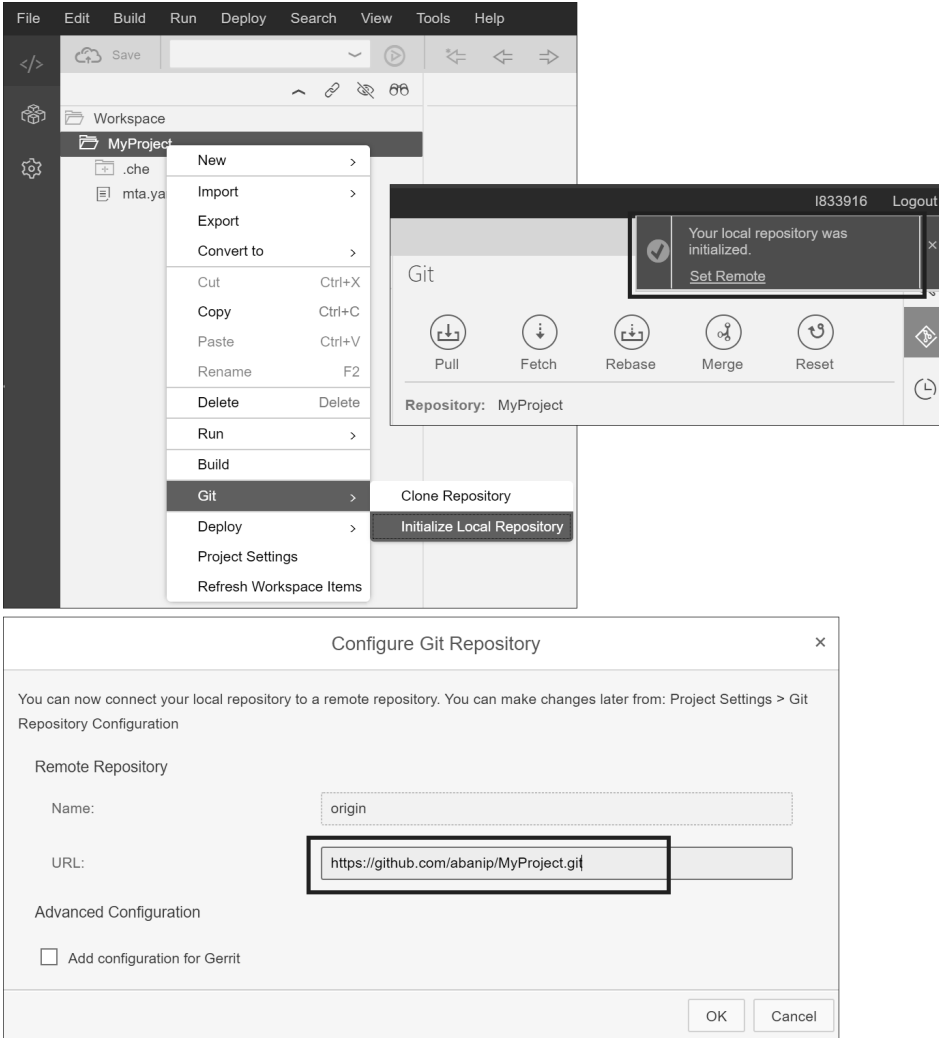


Figure 2.25 Setting Up the Git Repository

Note

Even though both the local and remote Git repositories are in synch at this point, it's a good practice to rebase the repository to establish proper synchronization between the local and remote repositories.

The **Git** pane of SAP Web IDE for SAP HANA has the following five buttons to work with the Git repository.

- **Fetch**
Downloads only new data/files from the remote repository, but it doesn't integrate any of this new data into the project working files. **Fetch** is great for getting the latest view of the remote repository, and it doesn't override or replace any files/objects of the local project.
- **Pull**
Downloads new data/files from the remote repository and integrates the new data into the current working files in the project. It's highly recommended to execute a **Pull** operation only after all local changes have been committed and pushed to the remote repository.
- **Rebase**
Merges another branch into the current working branch and moves all of the local commits that are ahead of the rebased branch to the top of the history on that branch.
- **Merge**
Takes two or more independent branches and integrates them into a single branch.
- **Reset**
Used to undo changes. There are two types of resets (**Mixed** and **Hard**). Both types update the head and index of the repository, while a **Hard** reset also resets the working directory.

After the development of the new object and files or changes to an existing object and file are complete, these changes can be saved in the central remote Git repository. The new files in the SAP HANA XS Advanced project are marked with + (plus), and changes are highlighted with * (asterisk), as illustrated in Figure 2.26 ❷.

The new files and changes to existing files are published to the remote repository using the following **Stage** and **Commit & Push** process (see Figure 2.26 ❶), and the log is displayed in the SAP Web IDE console (Figure 2.26 ❸).

After the changes are saved in the remote Git repository, the files in the local project reflect the status with the • (bullet) sign, and files will also be available in the remote Git repository (Figure 2.26 ❸ and ❹, respectively).

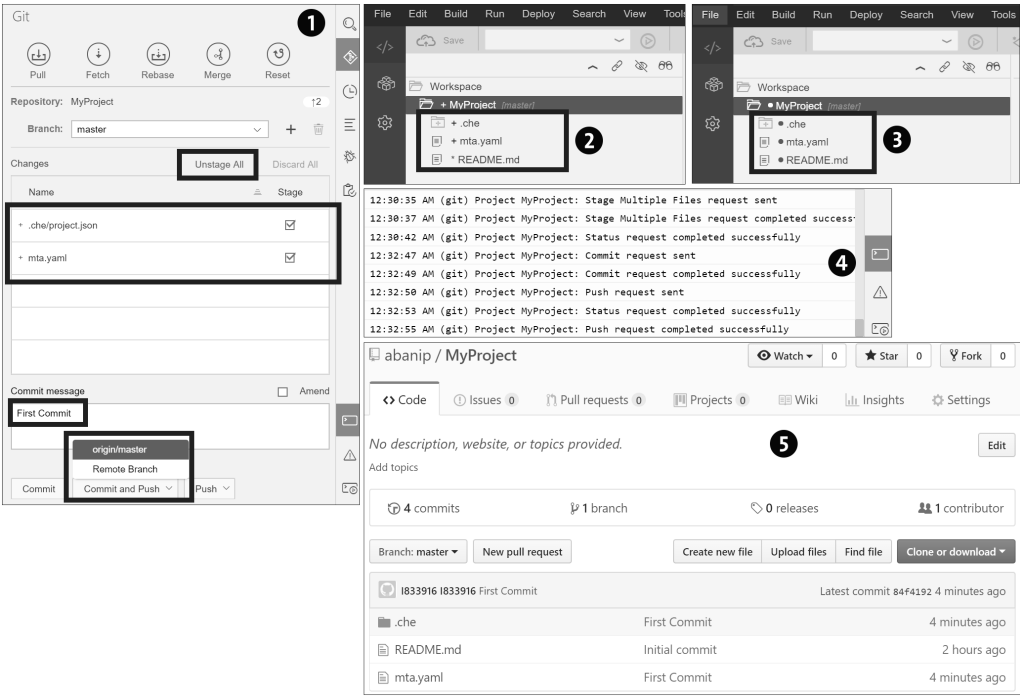


Figure 2.26 Git: Stage, Commit, Push

Finally, if another developer wants to work on your project or if you want to make changes for another project, the remote Git repository of the project can be cloned.

To create a project based on the Git repository, select **File • Git • Clone Repository**, or choose **Git • Clone Repository** in the context menu of the workspace as shown in Figure 2.27. Provide the Git repository URL, and select **Clone** to create the project. Upon successful cloning, the successful clone message will be displayed in the upper-right corner of SAP Web IDE. Select the **Create Local Branch** option to create the local repository for the project.

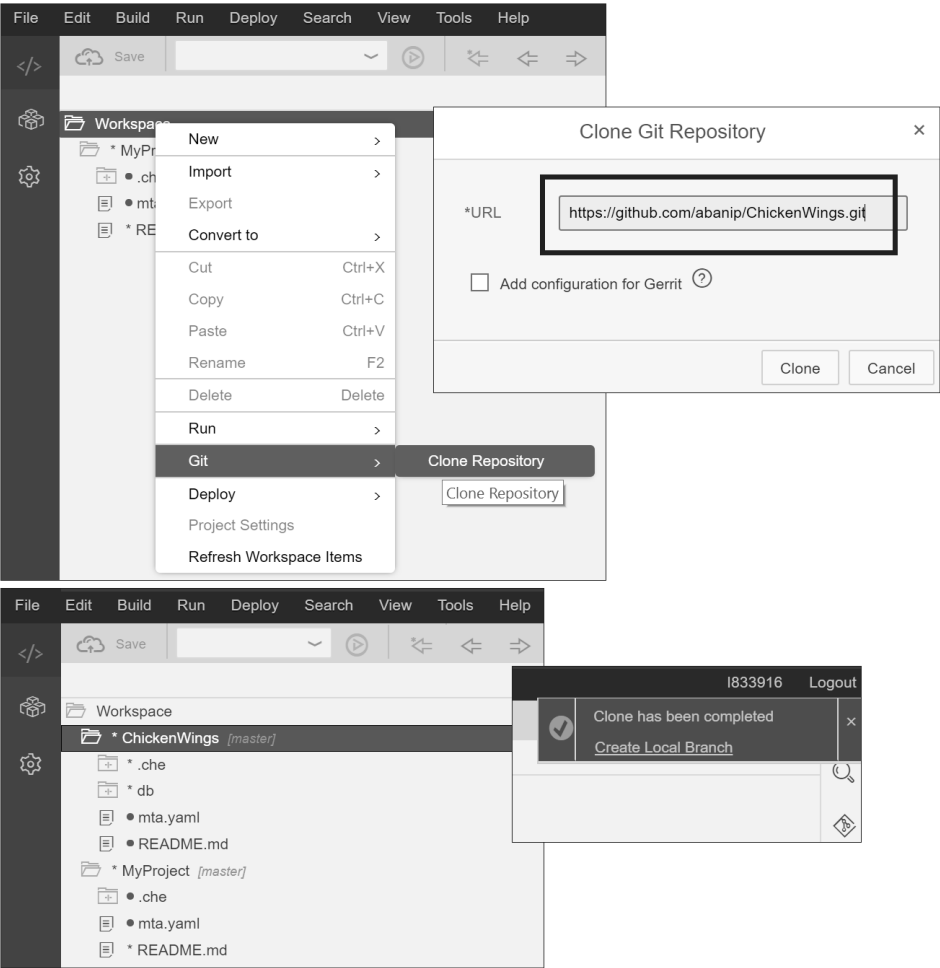


Figure 2.27 Cloning the Git Repository

2.7 Summary

We started this chapter with our discussion on the system landscape of traditional SAP systems and continued with the difference in the system landscape of SAP HANA XS Advanced to support continuous delivery and continuous integration.

Next, we discussed the installation and configuration of SAP HANA, express edition as a development environment in on-premise systems and the AWS public cloud.

You also learned about provisioning the SAP HANA system (including SAP HANA XS Advanced) as a service in the cloud using the SAP Cloud Platform.

We discussed the concept of organizations and spaces to share resources and isolate applications. Finally, we covered using Git as a source code management system and central code repository for SAP HANA XS Advanced development.

In the next chapter, we'll discuss development tools, including SAP HANA XS Advanced command-line interface (XSA CLI), SAP Web IDE for SAP HANA, SAP HANA database explorer, SAP HANA XS Advanced cockpit, SAP HANA Studio, and so on, to develop and administer SAP HANA XS Advanced applications.

Contents

Preface	15
---------------	----

PART I Getting Started

1 Introduction to the SAP HANA Development Platform	25
--	-----------

1.1 Business Cases for SAP HANA Applications	25
1.2 SAP HANA Components and Architecture	29
1.2.1 Functional Components	29
1.2.2 Technical Services and Architecture	36
1.3 Application Design Considerations	38
1.4 Summary	40

2 SAP HANA Development Environment	43
---	-----------

2.1 System Landscape	43
2.2 SAP HANA, Express Edition	47
2.2.1 On-Premise Installation Options	48
2.2.2 Installing SAP HANA, Express Edition VM Images	48
2.3 SAP Cloud Appliance Library	54
2.3.1 Registrations	55
2.3.2 SAP HANA, Express Edition Setup	56
2.4 SAP Cloud Platform	60
2.4.1 Registrations	61
2.4.2 Provisioning the SAP HANA Service Instance	61
2.4.3 Enable SAP Web IDE for Full-Stack Development	66
2.5 SAP HANA XS Advanced Organizations and Spaces	69
2.5.1 Organizations and Spaces Overview	69
2.5.2 Organization and Space Management	71

2.6	Git Code Repository	74
2.7	Summary	80
3	SAP HANA Development Tools	83
3.1	Command-Line Client	83
3.1.1	Logon and Setup	85
3.1.2	Application Management	86
3.1.3	Services Management	88
3.1.4	Organizations and Spaces	90
3.1.5	Domains	91
3.1.6	Certificates	92
3.1.7	Routes	93
3.1.8	Buildpacks	94
3.1.9	Runtime Environments and the Blob Store	95
3.1.10	Tasks	96
3.1.11	User Administration	97
3.1.12	Configuration	98
3.1.13	Plug-Ins	99
3.1.14	Other Commands	100
3.2	SAP Web IDE for SAP HANA	101
3.3	SAP HANA Database Explorer	104
3.4	SAP HANA XS Advanced Cockpit	109
3.5	Other SAP HANA Database Development Tools	113
3.5.1	SAP HANA Studio	113
3.5.2	SAP HANA Web-Based Development Workbench	115
3.5.3	SAP HANA Cockpit	116
3.5.4	Postman	117
3.6	Summary	118
4	SAP HANA XS Advanced Architecture	119
4.1	Microservices Architecture	119
4.1.1	Monolithic Architecture	120

4.1.2	Microservices Architecture	121
4.2	SAP HANA XS Advanced Architecture	122
4.2.1	12-Factor Apps	123
4.2.2	Cloud Foundry Basics	127
4.2.3	Organization of Applications, Services, and Users	129
4.2.4	The Controller	130
4.2.5	Deployment and Execution	131
4.2.6	Runtime Components and Services	133
4.2.7	Platform Services	135
4.3	SAP HANA XS Advanced Application Concepts	140
4.3.1	Multi-Target Application Concept	141
4.3.2	Architecture of a Multi-Target Application in SAP HANA XS Advanced	144
4.3.3	Development Workflow for Multi-Target Applications	146
4.4	Summary	147
 PART II Developing an Application		
5	Defining the Data Model	151
5.1	Data Model Overview	152
5.2	The Demo Application	154
5.3	Data Model Design in SAP HANA	159
5.3.1	Executing SQL Commands in SAP HANA	160
5.3.2	Commonly Used SQL Statements	163
5.3.3	Converting an Entity-Relationship Model to a Database Design	166
5.3.4	Data Model Using SAP HANA Repository Design-Time Objects	182
5.4	SAP HANA Deployment Infrastructure	189
5.4.1	HDI Containers	190
5.4.2	HDI Technical Users	191
5.4.3	HDI Deployer	192
5.4.4	Deployment into HDI Container	193
5.5	SAP HANA Database Module	196
5.5.1	Creating a Multi-Target Application Project	197
5.5.2	Creating an SAP HANA Database Module	199

5.5.3	Building the SAP HANA Database Module Artifacts	203
5.5.4	Building and Deploying a Multi-Target Application Archive	205
5.6	Core Data Services: The Physical Data Model	208
5.6.1	Editors	208
5.6.2	Entities	211
5.6.3	Data Types and User-Defined Structures	219
5.6.4	Associations	222
5.6.5	Views	224
5.6.6	Extensions	226
5.7	Loading Table Data	228
5.7.1	Using Table Data (.hdbtabledata)	229
5.7.2	Building the .hdbtabledata File	234
5.7.3	Using Table Data Properties (.properties)	235
5.7.4	Generate Time Data	236
5.8	Synonyms and Cross-Schema Access	237
5.9	Virtual Data Model	243
5.9.1	Create Calculation Views	245
5.9.2	Dimension Calculation Views	251
5.9.3	Default Calculation Views	263
5.9.4	Cube Calculation Views with Star Joins	264
5.9.5	Cube Calculation Views	272
5.9.6	Analytic Privileges	274
5.10	SQLScript for Stored Procedures	275
5.10.1	Stored Procedure	276
5.10.2	Table Functions and Scalar Functions	277
5.11	Table Creation without Core Data Services Documents	279
5.12	Other Database Artifacts	281
5.12.1	SAP HANA Database Sequence	281
5.12.2	SAP HANA Database View	282
5.12.3	SAP HANA Database Role	283
5.12.4	List of SAP HANA Deployment Infrastructure Artifacts	286
5.13	Summary	288

6	Developing the Application Layer	289
6.1	Tasks of the Application Layer	290
6.2	Node.js as Application Layer	292
6.2.1	Node.js Module	293
6.2.2	Modules and the Node Package Manager Repository	297
6.2.3	Asynchronous Programming Model	316
6.2.4	Managing Asynchronous Control Flow	320
6.2.5	Exposing Data	330
6.2.6	Accessing the SAP HANA Database	338
6.2.7	Unit Testing a Node.js Module	348
6.3	Java as Application Layer	358
6.3.1	Creating a Java Module to Read Database Content	360
6.3.2	Creating an OData Service to Modify Data	369
6.3.3	Connecting Java Services to an HTML5 Frontend	383
6.4	Summary	391
7	Developing a Presentation Layer	393
7.1	SAPUI5 Frontend Development	394
7.1.1	Application Bootstrapping	395
7.1.2	SAPUI5 Application Structuring	398
7.1.3	Model View Controller	401
7.1.4	Data Binding	404
7.1.5	Routing and Navigation	420
7.2	SAP HANA XS Advanced Application Routing	426
7.2.1	Application Routing Files	426
7.2.2	Application Routes and Destinations	429
7.2.3	SAPUI5 Central Service	431
7.3	Demo Application	436
7.3.1	Create the SAP Fiori Master-Detail Module	437
7.3.2	Demo Application Layout Adjustments	441
7.4	Summary	446

PART III Refining the Application

8	Securing Your Application	449
8.1	SAP HANA XS Advanced Security Concepts	450
8.2	Authorization in SAP HANA XS Advanced	455
8.2.1	Scopes and Attributes	455
8.2.2	Defining Application Security with the Application Security Descriptor	457
8.2.3	SAP HANA XS Advanced User Account and Authorization Service	463
8.2.4	Role Collections	464
8.2.5	Data Control Language	465
8.2.6	Controller Roles	466
8.3	Enable Security in your SAP HANA XS Advanced Application	467
8.3.1	Creating the SAP HANA XS Advanced User Account and Authorization Service	467
8.3.2	Enabling Security for Java Modules	470
8.3.3	Enabling Security for Node.js Modules	472
8.3.4	Enabling Security for SAPUI5 Modules	476
8.3.5	Secure Your Application against Web-Based Attacks	478
8.4	Maintaining Users and Roles in SAP HANA XS Advanced	482
8.4.1	Create Roles in SAP HANA XS Advanced	482
8.4.2	Create an Application User with the Administration Tools	487
8.4.3	Configuring Functional Authorization Checks	489
8.4.4	Configuring Instance-Based Authorization Checks	493
8.5	Assigning HDI Container Roles to Classic Database Users	495
8.5.1	Granting HDI Container Access via Schema Access	496
8.5.2	Granting HDI Container Access via Roles	498
8.6	Default Access Role for HDI Containers	500
8.7	Permissions for Container Objects	503
8.8	Summary	506

9	Troubleshooting Your Application	507
9.1	Debugging	507
9.1.1	Debugging Calculation Views	508
9.1.2	Debugging Stored Procedures	509
9.1.3	Debugging a Node.js Application	510
9.1.4	Debugging a Java Application	513
9.1.5	Debugging a SAPUI5 Application	515
9.2	Application Logs and Logging	519
9.2.1	Logging Data Changes	519
9.2.2	Logging Stored Procedure Actions	520
9.2.3	SAP HANA XS Advanced Transaction Logs	522
9.2.4	Logging in Node.js Applications	523
9.2.5	Logging in Java Applications	526
9.3	Database Traces	536
9.3.1	Configuring SQL Traces	537
9.3.2	Configuring Performance Trace	539
9.4	Performance Optimization	541
9.4.1	Execution of SQL Queries on Calculation Views	541
9.4.2	Analyze SQL in SAP Web IDE for SAP HANA	542
9.4.3	Explain Plan in SAP HANA Studio	547
9.4.4	Plan Visualizer in SAP HANA Studio	550
9.5	Summary	555
10	Deploying Your Application	557
10.1	Building the Multi-Target Application Archive	557
10.2	Deployment Process	560
10.2.1	Deployment via the SAP Web IDE for SAP HANA	560
10.2.2	SAP HANA XS Advanced Command-Line Interface	561
10.3	Transporting SAP HANA XS Advanced Applications	566
10.3.1	Manual Transport	566
10.3.2	Transport Using the Change and Transport System	567

10.3.3	Continuous Integration	568
10.3.4	Outlook: Native Transport	569
10.4	Summary	570

Appendices 571

A	Migrating an SAP HANA XS Application to SAP HANA XS Advanced	573
B	Additional Resources	593
C	The Authors	595
	Index	597

Index

@Data.publish annotation	360
12-Factor Apps	123
A	
<hr/>	
Access privileges	450
<i>HDI container</i>	451
ACID-compliant (atomicity, consistency, isolation, durability)	26
Admin processes	127
Administration application	487
Aggregation binding	407
<i>controller definition</i>	408
<i>defining within the controller</i>	413
<i>example</i>	410
<i>factory function</i>	412
<i>programmable</i>	411
<i>view definition</i>	409
Analysis view	243
Analytic privileges	274, 577
Analyze SQL tool	542–543
ANSI SQL	159
API key	565
Application concepts	140
Application deployment	557
Application descriptor	399
Application design	38
Application layer	289, 291
<i>tasks</i>	290
Application logs	519
Application monitor <i>viewing application logs</i>	522
Application package descriptor	294
Application privileges	590–591
Application programming interface (API)	31, 564
<i>container security</i>	473
Application Role Builder tool	464
Application router	135, 452, 477
<i>authorization checks</i>	454
<i>files</i>	426
Application Security Descriptor	457
approuter	426
Architecture	119, 122
<i>advanced</i>	453
<i>deploying and executing</i>	131
<i>microservices</i>	119
<i>organization</i>	129
<i>runtime components</i>	133
Async library	326
<i>further information</i>	330
Asynchronous function	320
<i>parallel processing</i>	321
Asynchronous programming model	316
Attribute view	577
Attributes	137, 152, 453, 456
<i>authorities</i>	460
<i>foreign-scope-references</i>	460
<i>oauth2-configuration</i>	461
<i>property</i>	459
<i>restricting</i>	484
<i>role-templates</i>	459
<i>tenant-mode</i>	462
Audit log	524
Audit log service for Java	531
<i>accessing logging entries</i>	535
<i>adding</i>	532
<i>creating log entries</i>	534
auditLog object	525
Audit-log service	139
Authentication token	455
Authorization	455, 581
<i>defining application security</i>	457
<i>scopes and attributes</i>	455
<i>setting up role templates</i>	456
<i>user account and authorization service</i> ...	463
Authorization checks	452
<i>configuring</i>	489
<i>configuring instance-based</i>	493–494
<i>declarative</i>	455
<i>instance-based</i>	454
<i>Java modules</i>	471
AWS	54
Azure	54

B

Backing services 124

Binary Linux installation 48

Binding context 406

Blob store 132–133

commands 95

Blocking 317

Bootstrapping 394–395

content delivery network 396

files 396

preload variant 397

Breakpoint

conditional 514

Java application 514

Node.js application 511

Build, release, run phases 125

Building MTAR 557–558

Buildpacks 132

commands 94

Business users 451

authorization checks 453

Business view 243

C

Calculated binding 418

Calculated field 418

Calculation column 261

Calculation views 33, 102

advanced properties 250

creating 245

data preview 250

debugging editor 508

properties 249

unfolding 542

Callback convention 307

Callback method 318

Cardinality 152

Cascading Style Sheets (CSS) 395

CDS associations 222

CDS editors 208

graphical editor 209

text editor 209

CDS entities 211

structure 211

CDS extensions 226

steps 227

CDS view 224

parameters 225

Certificates 92

Change and Transport System (CTS) 567

Click-jacking attacks 479

Cloud Foundry 60–61, 68

basics 127

environment 128

organization 129

setting up 61

Cloud Foundry Router 134

Codebase 123

Column tables 166

Columnar aggregation operations 167

Command-line interface (CLI) 71

Complex syntax 415

Component controller 399

definition 399

Concurrency 126

Configuration 124

commands 98

Container objects

permissions 503

Container schema 191

Content delivery network (CDN) 396

Continuous Integration 568

Control flow logic 291

Control template 410

Controller 402, 404

Controller roles 466

categories 466

resources 466

Core Data Services (CDS) 208, 360

CreateLog procedure 521

Cross-Origin Resource Sharing (CORS) 478

Cross-schema access 237

enabling 239

mta.yaml file 239

Cross-site request forgery (CSRF) 479

Cross-site scripting (XSS) 480

Cube calculation views 272

key features 267

Cube calculation views with star joins 264

input parameters 271

Cube calculation views with star joins (Cont.)

restricted columns 269

variables 270

Custom read method 382

D

Data binding 394, 402, 404

Data Control Language (DCL) 465

Data Definition Language (DDL) 31

Data extensions 275

Data foundation 265

Data model 152

defining 151, 167

demo application 154

Data model design 159

Data temperature 27

Database artifacts 281

deployment infrastructure 286

supported HDI artifacts 286

Database Explorer 160

Database explorer tool 509

Database layer 291

Database persistence 196

Database sequence 281

Database traces 536

Database view 282

Debugger panel 512, 515

Debugging 507

calculation views 508

enabling 510

integrated functionality 513

Java application 513

Java applications 514

Node.js application 510

SAPUI5 application 515

specifying breakpoints 511

stored procedures 509

Decision tables 575

Default calculation views 263

Delivery unit 183, 584

Denormalization 166

Dependencies 124, 301

Deploy app 305

Deploy MTAR 560–561

abort deployment 563

command line interface 561

Deploy MTAR (Cont.)

download deploy logs 564

restart deployment 563

retry deployment 563

SAP Web IDE 560

Deploy with namespaces 562

Deployed file system 193

Deployed runtime objects 182

Deployment extension descriptor 564

Deployment phases 131

Design-time artifacts 189

Design-time container (DTC) 190

Design-time definitions 182, 184

Destinations 135, 429

defining 430

Detail.view.xml 442

Dev/Ops 126

Dev/prod parity 126

Development (DEV) tier 43

Development environment 43

Development perspective 114

Development tools 83, 113

DevOps methodology 45

support 46

Dimension view 243, 251

column view 258

XML contents 255

Dispatcher 134

Disposability 126

Distinct count 269

Document Object Model (DOM) 395

Domain 91

Download Manager 49–50

Droplet 133

Droplets 132

E

Element binding 405

absolute model 407

programmatic 407

view definition 406

End user 452

accessing HTML5 application 491

assigning role collection 488

assigning roles 476

creating 487

End-to-end traces	537	GitHub	75
Enterprise information management		Global temporary tables	219
(EIM)	35	Google Chrome developer tools	515
Entities	152, 156	Grantable flag	458
Entity modifiers	217	Graph database features	34
Entity-relationship (ER) diagram	152	Graphical calculation view	577
Error messages		gulpfile.js	296
<i>forbidden</i>	472, 488		
Expensive statements traces	537	H	
Explain plan	547		
<i>columns</i>	547	Hash-based navigation	420
Express application		HDB module	196
<i>creating</i>	334	<i>artifacts</i>	202
Express framework	334	<i>building artifacts</i>	203
<i>application port listening</i>	337	<i>creating</i>	199–200
<i>further inforamtion</i>	338	<i>folder structure</i>	203
<i>mount middleware</i>	338	hdbgrants	503
<i>route path</i>	335	.hdbrole file	283
Express module	474	.hdbschema file	185
<i>adding as a dependency</i>	475	.hdbsequence file	188
Express security middleware	315	.hdbtable file	185
Expression binding	418, 420	.hdbtabledata	229
Extended store server	36	<i>building</i>	234
External buildpack processes	71	.hdbview file	187
External parse system	579	HDI API	193
		HDI client	194
F		HDI container	137, 189–190, 501
		<i>assigning roles</i>	502
Factory function	411	<i>database users</i>	501
Failed deployment	563	<i>default access role</i>	500
File-based module	298	<i>deployer</i>	192
File-system storage	139	<i>deployment</i>	193, 195
flight-rectoder.js	299	<i>granting access</i>	497
Flowgraph editor	202	<i>granting access via roles</i>	498
Folder-based module	298	<i>involved users</i>	451
Formatter function	415	<i>plug-ins</i>	195
Functional extensions	275	<i>proxy library</i>	195
		<i>roles</i>	495
G		<i>runtime database objects</i>	204
		<i>server</i>	195
Geo-spatial features	34	<i>technical users</i>	205
Gerrit	46, 74, 569	.hdiconfig file	280
Git	46, 74, 145–146	Helper function	374
Git repository	74, 78, 146, 291, 562, 569	Hierarchy	
<i>cloning</i>	79	<i>properties</i>	254
<i>git pane</i>	78	Hostname-based routing	134

HTML5	395	Job Scheduler	138
HTML5 module	384	Join Engine	578
<i>setting up router information</i>	386	Join order	577
Hypertext Transfer Protocol (HTTP)	31	JSON model	402
Hypervisors	49	JSON Web Token (JWT)	135, 315
I		K	
Identity provider (IdP)	450	Kernel profiler	537
index.html file	434	Key attributes	152
<i>adapting the bootstrap script</i>	434		
INF	153	L	
Information Access (InA)	31		
init method	400	Left outer join	246
Inner join	246	Level hierarchy	253
Intel Virtualization Technology	49	Lifecycle hooks	404
Internet of Things (IoT)	27	Logging	519
		<i>audit-logging</i>	524
J		<i>displaying log files</i>	522
		<i>fast changing data set</i>	520
Jasmine	348	<i>Java applications</i>	526
<i>expectations</i>	351	<i>last change</i>	519
<i>standard matchers</i>	351	<i>Node.js application</i>	523
Java	358	<i>severity level</i>	528
Java libraries	481	<i>slowly changing/moving dimension</i>	519
Java methods	372	<i>stored procedure actions</i>	520
Java modules		<i>table</i>	520
<i>creating</i>	360	<i>unsuccessful logon</i>	526
<i>creating a new class</i>	370	Logs	126
<i>creating in SAP Web IDE</i>	363		
<i>disabling security constraint</i>	366	M	
<i>generating</i>	364		
<i>running and testing</i>	377	Maintaining users and roles	482
<i>selecting</i>	363	Managed associations	222
<i>source code structure</i>	365	Managing asynchronous control flow	320
Java Naming and Directory		manifest.json	400
Interface (JNDI)	533	Manual transport	566
Java OData libraries	359	Master.controller.js	444
Java services		Master.view.xml	444
<i>API and service URLs</i>	383	Matcher function	351
<i>configuring binding information</i>	385	MDX console	105
<i>connecting to HTML5 frontend</i>	383	metadata property	400
<i>connecting to SAPUI5 table</i>	389	Microservices	32, 121
<i>creating HTML5 module</i>	384	Microservices architecture	119, 121
Java virtual machine (VM)	358	<i>advantages</i>	122
JavaScript	292, 395	Middleware	338

Migration	576	Neo	61, 66
Migration Assistant	575, 579, 585	Node database access	338
<i>intallation</i>	580	Node DB access	
<i>options</i>	586	<i>calling database procedure</i>	344
<i>report</i>	582, 585	<i>connection pooling</i>	341
Model	401	<i>database query</i>	343
Model view controller (MVC)	394, 401	<i>environment variables</i>	340
Modeled views	33	<i>Express middleware</i>	342
Modeler perspective	114	Node ID	254
Modules	297	Node.js	292, 299, 318
<i>notations</i>	300	<i>connecting</i>	338
<i>XSJS support</i>	314	<i>exposing data</i>	330
Monolithic architecture	120	Node.js event loop	319
MTA applications	431	Node.js module	293, 297, 437
MTA deployment descriptor	196	<i>debugging</i>	510
MTA editor		Node.js security packages	480
<i>defining destinations</i>	430	Node.js unit testing	348
mta.yaml	559	<i>choosing code to test</i>	354
mtad.yaml	559	<i>matchers</i>	351
.mtaext	564	<i>run unit test</i>	349
Multidimensional (cube) view	243	<i>test results</i>	349
Multidimensional expressions (MDX)	30	Normalization	153
Multi-target applications (MTA)	46, 66, 123, 196, 573	npm repository	297
<i>architecture</i>	144	<i>local npm repository</i>	301
<i>archive</i>	143, 145		
<i>archive builder</i>	559, 569	O	
<i>Build tool</i>	144	OAuth access token	136
<i>building an archive</i>	205–206	Object definition	162
<i>concept</i>	141–142	Object privileges	581, 590
<i>creating a project</i>	197–198	OData	31, 330
<i>deployer</i>	143	<i>associations</i>	332
<i>deploying an archive</i>	207	<i>deleting data</i>	381
<i>deployment descriptor</i>	143, 559	<i>exposing CDS definition</i>	360
<i>development descriptor</i>	142, 559	<i>functionality</i>	331
<i>development workflow</i>	146	<i>Java</i>	359
<i>node module</i>	293	<i>models</i>	402
<i>UAA</i>	142	<i>modifying data</i>	369
		<i>object name</i>	331
N		<i>parameters</i>	368
N and N+1 system landscape	44	<i>read service</i>	367
Name server	36	<i>service extension</i>	372
Namespace property	439	<i>updating data</i>	380
Native data types	217	One-time binding	405
Native transport	569	One-way binding	404
		Online analytical processing (OLAP)	27

Online transaction processing (OLTP)	27	Principle propagation	67
On-premise SAP HANA	128	Privileges	283
OpenUI5	397	<i>defining</i>	503
Operator list	546	<i>TRACE_ADMIN</i>	536
Organization	129	<i>with grant option</i>	504
<i>managing</i>	71	Procedural extensions	275
Organizations	69	Process	125
<i>cockpit</i>	111	Production (PRD) tier	43
<i>commands</i>	90	Projection node	252
		Promises	321
P		<i>further information</i>	326
Package privileges	581	Promisifying an API	323
package.json	294–295, 298, 427	.properties file	235
<i>scripts</i>	296	Property binding	414
Parallel processing	325, 328	<i>controller definition</i>	417
Parent-child hierarchy	254	<i>format options</i>	415
Path callback	336	<i>view definition</i>	416
Performance analysis tools	102	Protected parameters and properties	565
Performance optimization	541	PuTTY	51
<i>analyzing SQL</i>	542	PuTTYgen	59
<i>executing SQL queries</i>	541		
<i>explaining plan in SAP HANA Studio</i>	547	Q	
Performance traces	537	Quality (QAS) tier	43
<i>configuring</i>	539	Queueing service	124
<i>options</i>	539		
Persistence database	124	R	
Plan Visualizer	541, 550, 554	Referential join	247
<i>executed plan tab</i>	553	Reporting view	243
<i>overview tab</i>	551	Resource models	402
<i>using</i>	550	Return value	374
Platform controller	130	Right outer join	246
Platform services	135	Role collections	464, 483
Plug-ins		<i>assigning</i>	490
<i>commands</i>	99	<i>assigning application role</i>	486
Port binding	125	<i>assigning to end user</i>	487
Portal service	139	<i>creating</i>	485
Port-based routing	134	Role templates	484, 590
Postman	117, 378	Roles	283, 483
Postman REST client tool	378	<i>creating</i>	482
Preload modalities	397	<i>creating instance-based</i>	485
Presentation layer	291	<i>viewing</i>	483
<i>adding format option</i>	442	Route path	335
<i>changing the master list</i>	444	Router class	421
<i>developing</i>	393	<i>initializing</i>	421
<i>layout adjustments</i>	441		

Routes	422, 429	SAP HANA (Cont.)	
<i>commands</i>	93	<i>demo model</i>	582
<i>HTML5 module</i>	386	<i>development platform</i>	25
<i>matched event</i>	425	<i>in-memory database</i>	26
<i>navigating</i>	424	<i>integration services</i>	35
<i>pattern matched event</i>	425	<i>interfaces</i>	30
<i>properties</i>	429	<i>parsing system</i>	580
Routing	130	<i>platform</i>	30
Routing configuration	422	<i>platform innovations</i>	28
<i>defining parameters</i>	423	<i>programming model</i>	40
Rule-based optimizations	541	<i>technical services</i>	36
rungulp.js	296	SAP HANA administration console	114
Runtime container (RTC)	190, 291	SAP HANA cockpit	37, 116
Runtime tools	37	SAP HANA database container	106
S		SAP HANA database explorer	104, 106
SAP Cloud Appliance Library	54	SAP HANA Deployment Infrastructure	
<i>accessing SAP HANA, express edition</i>	59	(HDI)	37, 128, 189, 450, 590
<i>create instance</i>	56	<i>containers</i>	565, 587
<i>instance details</i>	58	<i>technical users</i>	191
<i>registrations</i>	55	SAP HANA development perspective	183
<i>SAP HANA, express edition post-</i>		SAP HANA Interactive Education package	
<i>configuration</i>	58	(SHINE)	582, 588
<i>SAP HANA, express edition setup</i>	56	SAP HANA Platform cockpit	64
SAP Cloud Platform	60	SAP HANA Repository	44, 74, 182
<i>registrations</i>	61	<i>design-time objects</i>	182
SAP Cloud Platform cockpit	63	SAP HANA Rules Framework	575
SAP Enterprise Performance Management ...	34	SAP HANA Secure Store	138
SAP Event Stream Processor	35	SAP HANA Service Broker	137, 191
SAP Fiori	393	SAP HANA service Instance	
SAP Fiori launchpad	139	<i>provisioning</i>	61
SAP Fiori Master-Detail Module	437	SAP HANA service instance	60
<i>binding OData properties</i>	440	<i>creating</i>	62
<i>creating</i>	437	SAP HANA services	137
<i>running</i>	441	SAP HANA smart data integration (SDI)	35
<i>selecting OData service</i>	438	SAP HANA smart data quality (SDQ)	35
<i>testing</i>	440	SAP HANA Studio	113, 574
SAP HANA	159	<i>accessing Cloud Foundry</i>	65
<i>application business cases</i>	25	<i>creating design-time objects</i>	183
<i>application layers</i>	39	<i>database objects</i>	162
<i>application services</i>	32	EXPLAIN PLAN	547
<i>architecture</i>	29, 38	<i>migration tool</i>	575
<i>cloud-based deployment</i>	27	<i>migration type</i>	576
<i>components</i>	29	SAP HANA views	481
<i>database</i>	39	SAP HANA Web-Based Development	
<i>database services</i>	32	Workbench	37, 115
		<i>functionalities</i>	115
		<i>traces</i>	536

SAP HANA XS	53, 122, 182, 569, 574	SAP Web IDE for SAP HANA	37, 560, 579
<i>artifacts</i>	587	sap.app namespace	400
<i>migration</i>	573	sap.ui namespace	401
SAP HANA XS Advanced	400	sap.ui5 namespace	401
<i>application routing</i>	426	SAPUI5	393
<i>migration</i>	573	<i>application structuring</i>	398
<i>overview</i>	32	<i>central service</i>	431
<i>SAPUI5 runtime</i>	431–432	<i>debugging tools</i>	517
SAP HANA XS Advanced Admin tool	109	<i>frontend development</i>	394
SAP HANA XS Advanced cockpit	109	<i>routing and navigation</i>	420
<i>displaying applications</i>	112	<i>SmartTable</i>	390
<i>functionalities</i>	110	Scalar functions	277–278
<i>opening</i>	111	Schema	138
SAP HANA XS Advanced cockpit web		Schema access	496
<i>application</i>		Scopes	136, 453, 455
<i>creating end users</i>	488	<i>defining</i>	458
<i>roles</i>	482	<i>property</i>	458
SAP HANA XS JavaScript (XSJS)	587	<i>viewing</i>	483
SAP HANA, express edition	47	Search and text analysis	33
<i>checking application status</i>	54	Security	449, 587
<i>excluded features</i>	47	<i>application access</i>	588
<i>installing VM images</i>	48	<i>concepts</i>	450
<i>on-premise installation</i>	48	<i>enabling</i>	467
<i>organization and space</i>	71	<i>Java modules</i>	470
<i>post-installation checks</i>	52	<i>Node.js modules</i>	472
<i>prerequisites</i>	49	<i>roles and privileges</i>	589
SAP Node.js modules	303	<i>SAPUI5 modules</i>	476
@sap/hdi-deploy	305	<i>web-based attacks</i>	478
@sap/textanalysis	305	Security Assertion Markup Language	
@sap/textbundle	308	(SAML) 2.0 standard	452
@sap/xsenv	310	Security constraint	
@sap/xsjs	312	<i>enabling</i>	471
@sap/xssec	314	Semantic versioning	300
<i>application environment variables</i>	310	Semantics	266
<i>application router</i>	303	Sequential processing	324, 326
<i>container security API</i>	314	server.js	296
<i>database client</i>	304	Server-side application	120
HDI Deployer	305	Service broker	70, 130, 136
<i>internationalization</i>	308	Service instance	130
SAP HANA Database Client	304	Service plan	129
SAP HANA XS compatibility layer	312	Service query object	310
SAP HANA-based text analysis	305	Services	88, 129
SAP Web IDE	60, 101, 508	Simple Logging Facade for Java (SLF4J)	527
<i>accessing</i>	59, 102	<i>initializing</i>	527
<i>creating Java module</i>	362	<i>logging information</i>	530
<i>full-stack development</i>	66	<i>logging levels</i>	528
<i>preferences</i>	103	<i>removing logging levels</i>	530

Single-page pattern	395
Smart controls	394
Software Development Kit (SDK)	395
Source code	145–146
Space Enablement tool	73
Spaces	69–70, 129, 466
<i>adding users</i>	73
<i>cockpit</i>	112
<i>commands</i>	90
<i>creating</i>	72
<i>enabling</i>	73
<i>managing</i>	71
SQL	32, 159
SQL analytic privileges	493
SQL analyzer	105
SQL commands	
<i>executing</i>	160
SQL Console	37, 105, 160
SQL Console (Admin)	496
SQL debugger	105
SQL Engine	541
<i>execution plans</i>	547
SQL executor	541
SQL optimizer	541
SQL processor	194
SQL queries	541
<i>dominant operators</i>	543
<i>optimizing behaviors</i>	541
<i>plan graphs</i>	544
SQL statements	163
<i>ALTER TABLE</i>	165
<i>COMMENT ON</i>	164
<i>CREATE SCHEMA</i>	163
<i>CREATE TABLE</i>	164
<i>INSERT INTO</i>	165
<i>SELECT</i>	166
<i>SET SCHEMA</i>	163
SQL traces	536
<i>configuring</i>	537
<i>levels</i>	538
<i>options</i>	537
SQLScript	32, 102
<i>best practices</i>	275
<i>stored procedures</i>	275
Staging process	133
Star join	264–265
Stored procedure	
<i>loading and calling</i>	345
Stored procedures	276
Structure data type	220
Structured Query Language (SQL)	30
Synchronous programming model	316
Synonym editor	238
Synonyms	237, 503
System database	62
System landscape	43, 46
System privileges	581
T	
Table accesses view	545
Table data	
<i>attributes</i>	231
<i>generating time data</i>	236
<i>loading</i>	228
<i>using plug-in</i>	229
<i>using properties</i>	235
Table functions	277
Tables	
<i>AIRPORT</i>	168
<i>BOOKING</i>	179
<i>CONNECTION</i>	175
<i>CREW</i>	173
<i>CUSTOMER</i>	171
<i>FLIGHT</i>	177
<i>PLANE</i>	169
<i>PRICE_CALENDAR</i>	174
<i>SEAT</i>	175
Tables in use view	545
Technical users	451
Tenant database	62
Test driven development	354
Text analysis	
<i>example</i>	308
<i>parameters</i>	306
Text bundles	308
Text join	247
Time data	236
<i>generating</i>	236
Timeline view	545, 554
Timeouts	
<i>configuring</i>	480

Tomcat Java runtime	533
Transaction logs	522
Transport using CTS	567
Transporting SAP HANA XS Advanced applications	566
Troubleshooting	507
Two-way binding	405

U

Undeploy MTA	565
<i>delete services</i>	566
undeply.json	204
Uniform Resource Identifier (URI)	372
Union node	273
Unit testing	348
Unmanaged associations	223
Upstream repository	302
User acceptance (UAT) tier	43
User Account and Authentication	
Service (UAA)	135–136, 314, 463, 565
<i>binding</i>	476
<i>creating</i>	467
User Management tool	464
<i>migrating SAP HANA user</i>	488
User-defined data types	220
User-defined structures	220
User-provided services	140, 144
Users	70, 129
User-specific traces	536

V

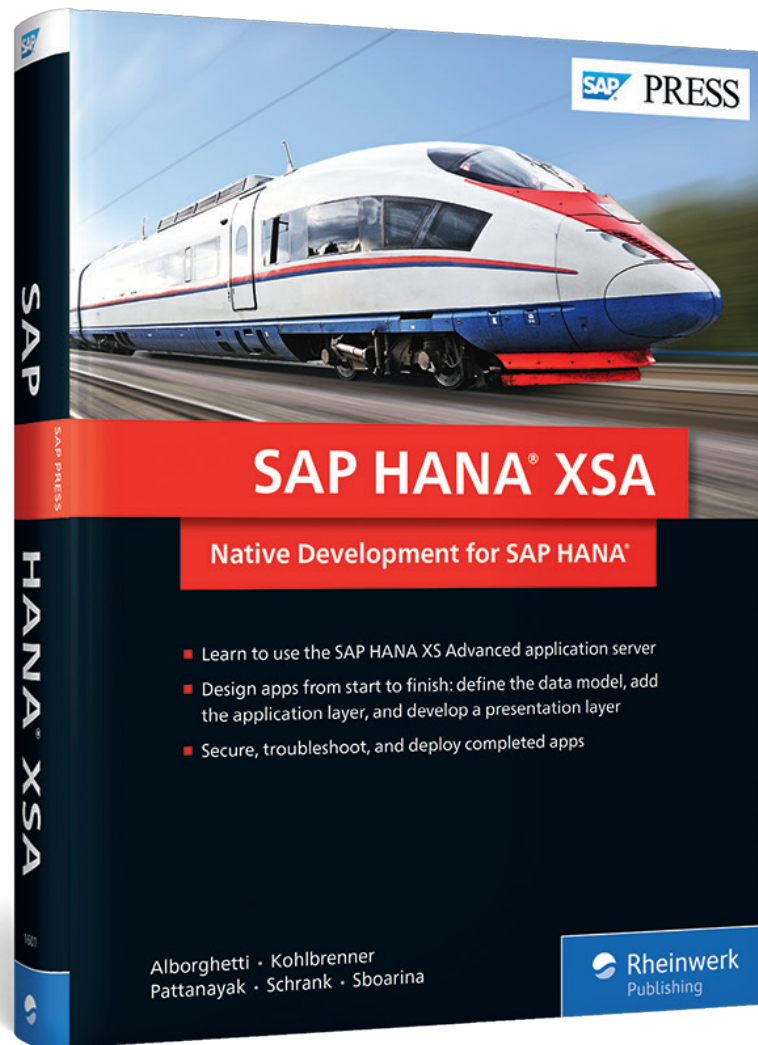
View	401, 403
<i>JavaScript</i>	403
Virtual data models	243
VM image	48

W

WebSocket communication	429
Work file system	193

X

XML for Analysis (XMLA)	31
XML namespaces	403
XS Engine	122
XS runtime	36
XSA CLI	83, 529, 561–562
<i>application management</i>	86
<i>buildpacks</i>	94
<i>certificates</i>	92
<i>configuration</i>	98
<i>domains</i>	91
<i>getting</i>	84
<i>help functionality</i>	84
<i>organizations and spaces</i>	90
<i>other commands</i>	100
<i>plug-ins</i>	99
<i>routes</i>	93
<i>runtime environments</i>	95
<i>services management</i>	88
<i>setting up</i>	85
<i>tasks</i>	96
<i>user administration</i>	97
xs-app.json	427
<i>sections</i>	428
XSASecurityContext functions	473
xsenv module	310
XSODATA	330
<i>associations</i>	333
<i>further information</i>	333
XSODATA service	436
xs-security.json	457, 468
<i>xsappname property</i>	457



Francesco Alborghetti, Jonas Kohlbrenner, Abani Pattanayak, Dominik Schrank, Primo Sboarina

SAP HANA XSA: Native Development for SAP HANA

607 Pages, 2018, \$79.95
ISBN 978-1-4932-1601-7

 www.sap-press.com/4500



Francesco Alborghetti is the cofounder of and a managing partner for Inquaero, an innovative consulting company with a focus on SAP lean data management solutions. He is an expert SAP consultant who has developed his wide expertise for more than 15 years.



Jonas Kohlbrenner is a senior SAP HANA engineer and architect. His areas of focus include SAP HANA Basis technology topics like data center integration and operations, SAP HANA application architecture and development, as well as SAPUI5 application architecture design and development.



Abani Pattanayak has more than 15 years of experience in data warehousing, and has worked with SAP in-memory technology since 2007. As a principal architect at SAP and an SAP HANA Distinguished Engineer, he has helped implement and performance-tune SAP HANA for multiple customers.



Dominik Schrank is a principal consultant for SAP HANA and SAP analytics solutions at SAP Deutschland SE & Co. KG. He has more than ten years of experience in designing and implementing large business analytics and native SAP HANA applications.

Primo Sboarina has worked in the SAP environment since 2001 and joined SAP in 2007. His focus is on product development for evolving SAP technologies and on data management. He is currently focused on promoting and supporting SAP S/4HANA.

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.