INTERNAL

Demo of the 12 Factor methodology in BTP Cloud Foundry

This document will guide you step by step to deploy an app that demonstrates the 12 Factor methodology in BTP Cloud Foundry

www.sap.com/contactsap

THE BEST RUN SAP®

Table of Contents

DISCLAIMER

The information shared in this document is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. All functionality presented here is subject to change and may be changed by SAP at any time for any reason without notice.

OBJECTIVE

This document is aimed for beginners.

The objective of this exercise is to demonstrates the 12 Factor methodology in BTP Cloud Foundry.

Several concepts like Cloud Foundry Spaces, Cloud Foundry Marketplace and many Cloud Foundry commands will be explained, and a sample source code will be provided.

SCENARIO

This exercise demonstrates the 12 Factor methodology in BTP Cloud Foundry using a Node.js app.

PREREQUISITES

- SAP BTP Trial Account - Get a Free Account on SAP BTP Trial
- Enable **Cloud Foundry Environment** and create an Organization and a Space – This is done automatically for you when you get a Free Account on SAP BTP trial - Creating a Cloud Foundry Organization and Space

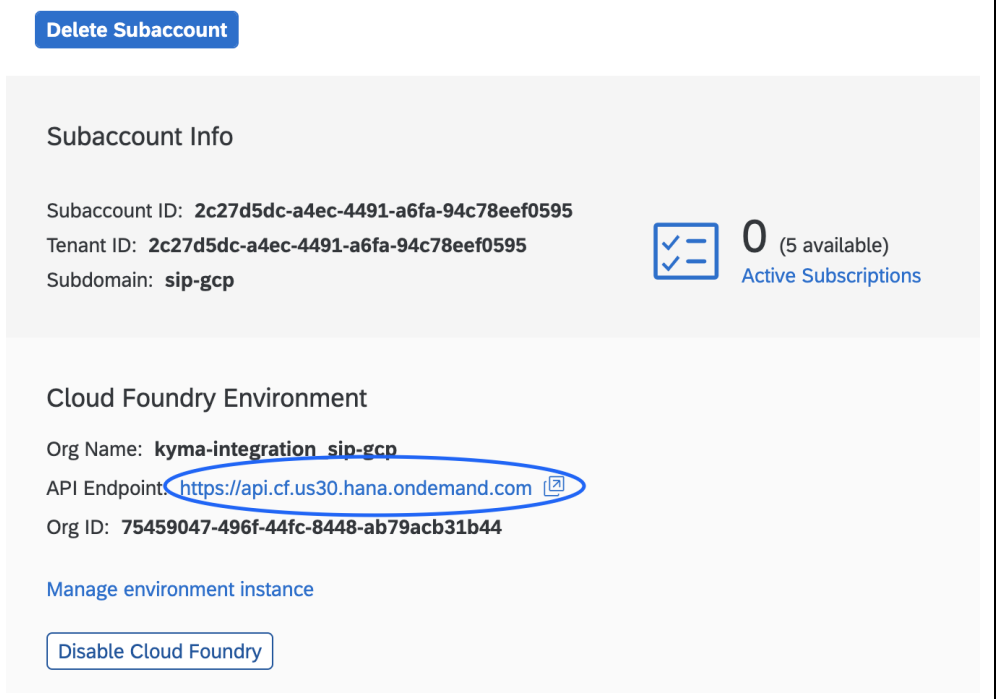
SUGGESTED PRE-READS

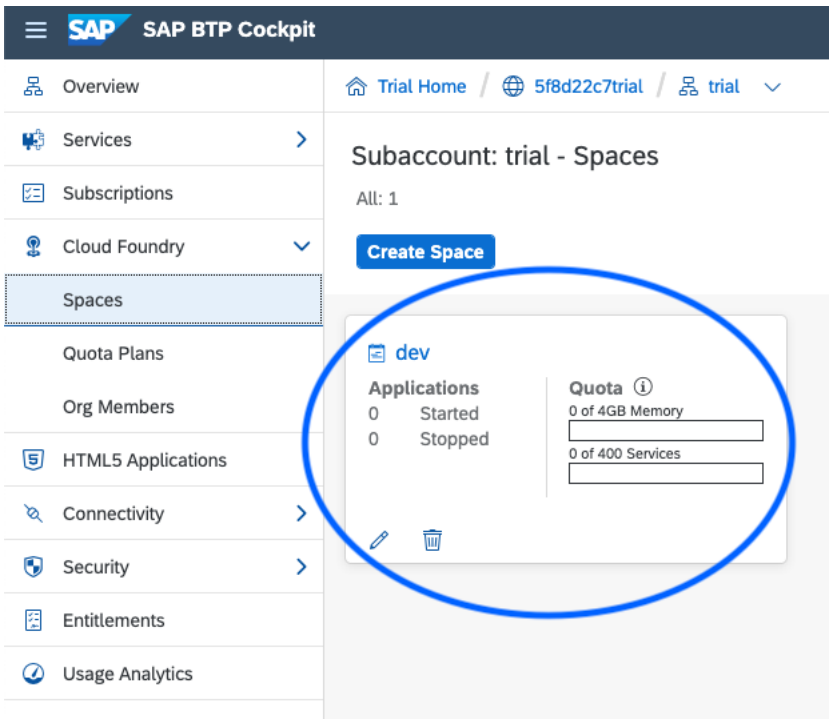
- Get Ready to Develop on SAP BTP
- Getting Started in the Cloud Foundry Environment
- Development in the Cloud Foundry Environment
- Developing Node.js in the Cloud Foundry Environment
- Install the Cloud Foundry Command Line Interface (CLI)
- Creating a Cloud Foundry Organization and Space

SOURCE CODE REPOSITORY

- https://github.com/Ralphive/btp-12-dogs-app/

DEPLOYING THE APPLICATION – FROM THE TERMINAL / COMMAND PROMPT

Explanation	Screenshot
Clone the GitHub repository Clone the GitHub repository using the following command: `git clone https://github.com/Ralphive/btp-12-dogs-app/`	
Login to Cloud Foundry The first step is to login to Cloud Foundry. When using the cf login command, the cf CLI prompts for credentials as needed. If you are a member of multiple orgs or spaces, cf login prompts you for which ones to log in to. Otherwise, it targets your org and space automatically. Run the following command: *cf login -a <API-endpoint>* The API endpoint can be retrieved from the BTP subaccount. It is listed under the Cloud Foundry Environment section as shown in the screenshot on the right. **Example:** `cf login -a https://api.cf.us30.hana.ondemand.com`	

Explanation	Screenshot
When you get a free Account on SAP BTP Trial, an **org** and a **space** with the name “dev” are created for you automatically.	
Create a Cloud Foundry Space **Note:** Please follow this step only if you don’t have a space, or if you want to use a new space. Use cf create-space {space_name} to create a new space within your Org. Create a Cloud Foundry Space by using the following command. *cf create-space <Space-Name>* **Example:** cf create-space dev	[\W $cf create-space dev
Target the new space *cf target -o ORG -s SPACE* Example:	[\W $cf target -o "kyma-integration_sip-gcp" -s "dev"

Explanation	Screenshot
cf target -o "kyma-integration_sip-gcp" -s "dev"	
Push the app to Cloud Foundry The cf CLI command cf push pushes apps to CF. There are two main ways to run the cf push command: • Run cf push APP-NAME to push an app the easiest way, using default settings. • Run the cf push command with flags and helper files to customize: a. How the pushed app runs, including its route (URL), instance count, and memory limits. b. How the push process works: whether it's configured with a manifest, runs a startup script, or limits files uploaded to the Cloud Controller. For more information, please refer to this document. Navigate to the directory you cloned and push the app to Cloud Foundry. cf push --random-route	\W $cf push --random-route

Explanation	Screenshot
The app will be deployed and started. You can now access it via the route displayed in the terminal. Example: https://12-dogs-bogus-raven-el.cfapps.us30.hana.ondemand.com	Waiting for app 12-dogs to start... Instances starting... Instances starting... Instances starting... Instances starting... name: 12-dogs requested state: started routes: 12-dogs-bogus-raven-el.cfapps.us30.hana.ondemand.com last uploaded: Tue 08 Jun 15:55:38 EDT 2021 stack: cflinuxfs3 buildpacks: **name** **version** **detect output** **buildpack name** https://github.com/cloudfoundry/nodejs-buildpack.git 1.7.53 nodejs nodejs type: web sidecars: instances: 1/1 memory usage: 128M start command: npm start **state** **since** **cpu** **memory** **disk** **details** #0 running 2021-06-08T19:55:51Z 0.0% 0 of 0 0 of 0
Set environment variables The user-provided variables set using the cf set-env command Set the environment variables using the following commands. cf set-env 12-dogs DOG_BREED spaniel cf set-env 12-dogs DOG_SUBBREED cocker	[\W $cf set-env 12-dogs DOG_BREED spaniel Setting env variable DOG_BREED for app 12-dogs in org kyma-integration_sip-gcp OK TIP: Use 'cf restage 12-dogs' to ensure your env variable changes take effect. [\W $cf set-env 12-dogs DOG_SUBBREED cocker Setting env variable DOG_SUBBREED for app 12-dogs in org kyma-integration_sip-gcp OK TIP: Use 'cf restage 12-dogs' to ensure your env variable changes take effect.
Restage Your App To restage your app, run the following command: *cf restage YOUR-APP* Restaging your app stops your app and restages it, by compiling a new droplet and starting it. Restage your app if you have changed the environment in a way that affects your staging process, such as setting an environment variable that	[\W $cf restage 12-dogs This action will cause app downtime. Restaging app 12-dogs in org kyma-integration_sip-gcp

Explanation	Screenshot
the buildpack consumes. The staging process has access to environment variables, so the environment can affect the contents of the droplet. You should also restage your app whenever you edit any configuration settings, such as when you rename it, add metadata, or configure health checks. The new settings often do not take effect until you restage the app. Restaging your app compiles a new droplet from your app without updating your app source. Use 'cf restage 12-dogs' to ensure your env variable changes take effect. Example: `cf restage 12-dogs`	
All instances of the app are restarted when restaging the app.	 Stopping app... Waiting for app to start... Instances starting... Instances starting... Instances starting... name: 12-dogs requested state: started routes: 12-dogs-bogus-raven-el.cfapps.us30.hana.ondemand.com last uploaded: Tue 08 Jun 15:59:54 EDT 2021 stack: cflinuxfs3 buildpacks: **name** **version** **detect output** **buildpack name** https://github.com/cloudfoundry/nodejs-buildpack.git 1.7.53 nodejs nodejs type: web sidecars: instances: 1/1 memory usage: 128M **state** **since** **cpu** **memory** **disk** **details** #0 running 2021-06-08T20:00:06Z 1.2% 30.6M of 128M 119.7M of 1G

List Marketplace Services
 	
After targeting and logging into Cloud Foundry, run the `cf marketplace` cf CLI command to view the services available to your
	`\W $cf marketplace`
	
	`\W $cf marketplace -e postgresql-db`

Explanation	Screenshot

targeted organization. Available services may differ between organizations and between Cloud Foundry marketplaces.
 	
Get the list of service offerings that are available in the service marketplace by using the following command.
 	
`cf marketplace`
 	
You'll notice the following PostgreSQL service offering in the list that is returned.
 	
postgresql-db trial PostgreSQL service on SAP BTP
 	
Use the following command to view descriptions of individual plans of a given service offering.
 	
cf marketplace -e SERVICE_OFFERING
 	
Example:
 	
`cf marketplace -e postgresql-db`
 	
In the next step, create an instance of this service using one of its plans.
	plan **description** **free or paid** **costs** **available**
	trial Trial PostgreSQL service offering free yes

Creating a Service Instance You can create a service instance with the following command: cf create-service SERVICE PLAN SERVICE_INSTANCE	`$cf create-service postgresql-db trial dog-db`

Explanation	Screenshot
Use the information in the list below to replace SERVICE, PLAN, and SERVICE_INSTANCE with appropriate values. • SERVICE: The name of the service you want to create an instance of. • PLAN: The name of a plan that meets your needs. Service providers use plans to offer varying levels of resources or features for the same service. • SERVICE_INSTANCE: The name you provide for your service instance. You use this name to refer to your service instance with other commands. Service instance names can include alpha-numeric characters, hyphens, and underscores, and you can rename the service instance at any time. Create the DB service Instance using the following command. *cf create-service <Service Name> <Service Plan> <Instance Name>* **Example:** cf create-service postgresql-db trial dog-db	
Bind the Service Instance to the App Depending on the service, binding a service instance	\W $cf bind-service 12-dogs dog-db Binding service dog-db to app 12-dogs in org kyma-integration_sip-gcp

Explanation	Screenshot
to your app may deliver credentials for the service instance to the app. See the Delivering Service Credentials to an App topic for more information. Binding a service instance to an app may also trigger app logs to be streamed to the service instance. For more information, see Streaming App Logs to Log Management Services. Bind the service instance with the application using the following command. *cf bind-service <Application-Name> <Service-Instance-Name>* **Example:** `cf bind-service 12-dogs dog-db`	

CONFIGURE PSTGRESQL IN CLOUD FOUNDRY FOR EXTERNAL ACCESS

When deploying the app locally, PostgreSQL and be configured in Cloud Foundry for external access via the steps listed below.

First, create a service key to access the PostgreSQL service from outside Cloud Foundry.

cf create-service-key <SERVICE_INSTANCE> <SERVICE_KEY>

```
cf create-service-key dog-db db-key
```

Then, check the PostgreSQL credentials. User, Password, IP and DB Name.

cf service-key <SERVICE_INSTANCE> <SERVICE_KEY>

Example:

```
cf service-key dog-db db-key
```

The above command will return a response along with the **dbname**, **hostname**, **password**, **port** and other details.

Then, create a secure SSH tunnel from your local system to the remote PostgreSQL.

cf ssh -L <local-port>:<service-IP>:<service-port> <your-app>

Example:

```
cf ssh -L 63306:10.11.241.35:60262 12-dogs
```

CONNECT TO POSTGRESQL AND RUN A SCRIPT IN THE CLOUD

Now connect to PostgreSQL from another terminal.

psql -d <DatabaseName> -U <UserName> -p <LocalPort> -h localhost

psql -d zTumYIKsJnai -U e30c278d60b4 -p 63306 -h localhost

or

\c <DatabaseName> <UserName> localhost <LocalPort>
\c zTumYIKsJnai e30c278d60b4 localhost 63306

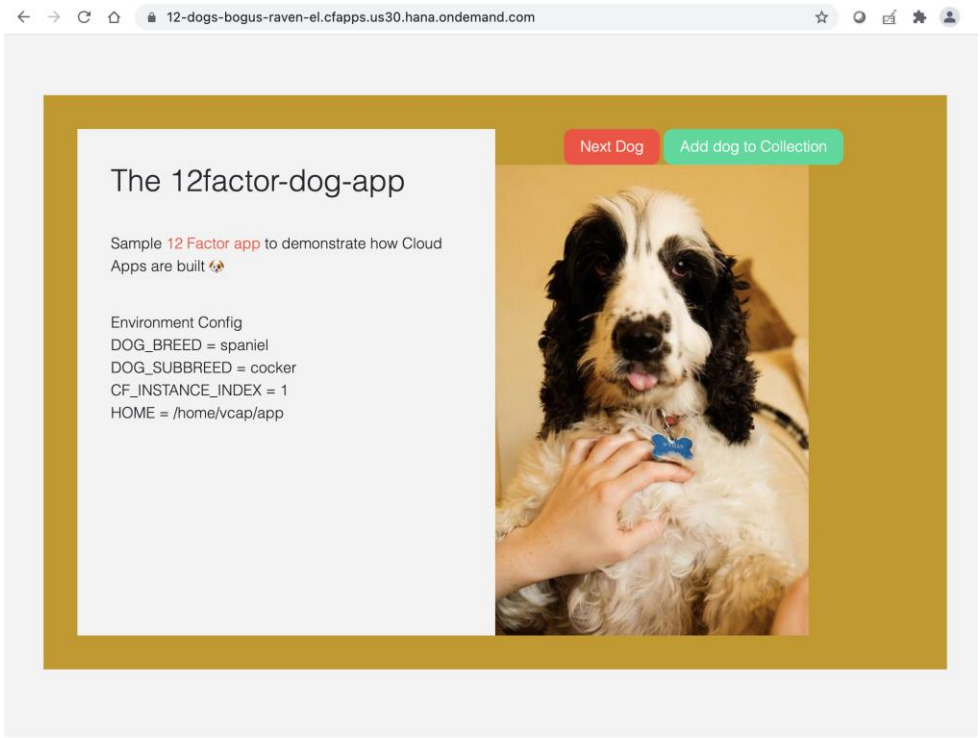
After inserting the password, you can list all the DBs.

\l

And, create the table that the app needs by running the code that is within the following file: /db/initialize.sql

SELECT * FROM dog_collection

FINAL STEPS

Explanation	Screenshot
Restart Your App To restart your app, run the following command: *cf restart YOUR-APP* **Example:** cf restart 12-dogs	[\W $cf restart 12-dogs Restarting app 12-dogs in org kyma-integration_sip-gcp
Access you App Run the app on the route displayed in the terminal. **Example:** https://12-dogs-bogus-raven-el.cfapps.us30.hana.ondemand.com	

SCALING THE APP

Scaling Horizontally

Horizontally scaling an app creates or destroys instances of your app.

Incoming requests to your app are automatically load balanced across all instances of your app, and each instance handles tasks in parallel with every other instance. Adding more instances allows your app to handle increased traffic and demand.

Use the following command to horizontally scale your app. Cloud Foundry will increase or decrease the number of instances of your app to match INSTANCES.

cf scale <your app> -i <# instances>

Example:

```
cf scale 12-dogs -i 3
```

```
cf app 12-dogs
```

Reference: Scaling an App Using cf scale