Deploying applications to Cloud Foundry on IBM Cloud

Unit objectives

- Explain how to manage your IBM Cloud account with the IBM Cloud CLI.
- Describe how to create a Node.js application that runs on IBM Cloud.
- Deploy an application from a local workstation by using the IBM Cloud CLI.
- Describe the role of Node.js for server-side scripting.
- Deploy an application by using IBM Cloud App Service (Web Apps).

Introduction to Cloud Foundry

Topics

- ▶ Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- Resiliency
- Logging and debugging
- Domains and routes
- Binding external services
- Next steps

Cloud Foundry benefits

- Helps you deploy applications.
- Decouples applications from infrastructure.
- Language- and framework-neutral.
- Makes building, deploying, and scaling apps fast and easy.
- Is an open cloud-native platform.

IBM Cloud provides a **certified** Cloud Foundry platform.

Deploying Cloud Foundry applications with IBM Cloud CLI

Topics

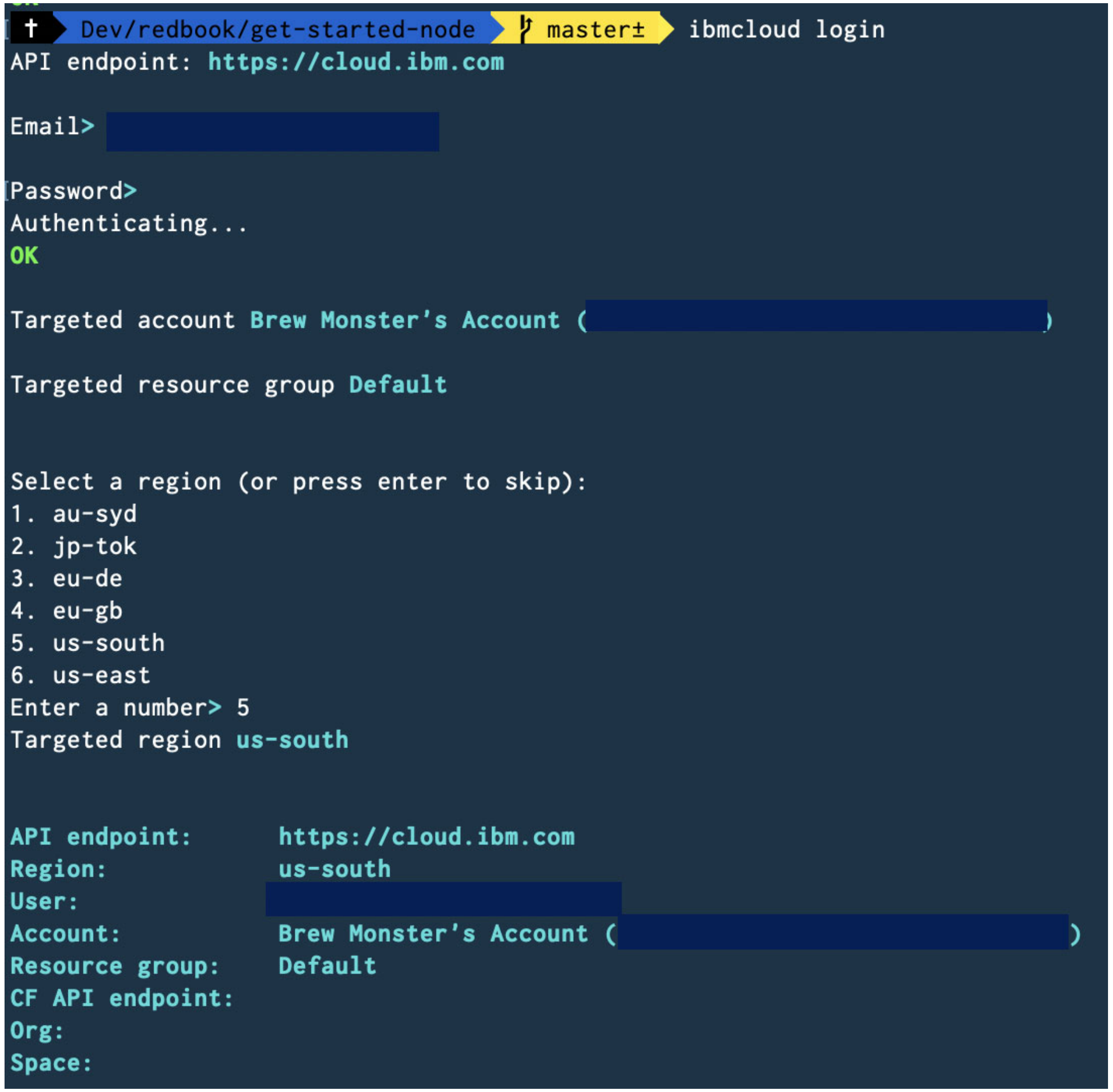
IBM Cloud CLI overview

- IBM Cloud CLI is a general-purpose developer tool that provides access to an IBM Cloud account and services through a command-line interface (CLI).
- Cloud Foundry commands are accessible by using **`ibmcloud cf`**.
- For a complete list of available commands, run
  `ibmcloud cf help -a`
- For the latest installer and instructions, go to:
  https://cloud.ibm.com/docs/cli

Deploying your first Node.js application – Prerequisites: Prepare IBM Cloud

- Before you begin, verify that:
  - You have access to an IBM Cloud account.
  - The IBM Cloud CLI installed is installed in your workstation.
- Run `ibmcloud login` and follow the instructions.
- You are prompted to enter your email, password, and a region to deploy your application.

Deploying your first Node.js application - Prerequisites: Prepare IBM Cloud (cont.)



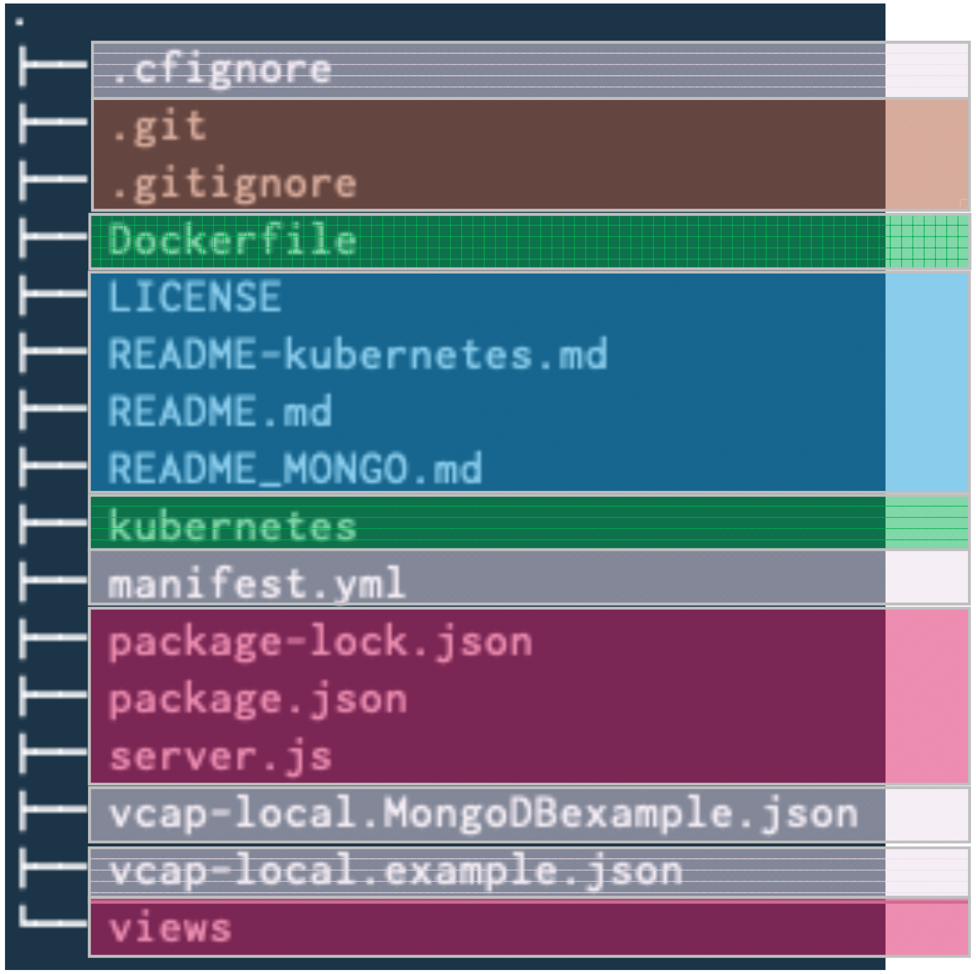
```
✝ Dev/redbook/get-started-node  master± ibmcloud login
API endpoint: https://cloud.ibm.com

Email> [redacted]

Password>
Authenticating...
OK

Targeted account Brew Monster's Account ([redacted])

Targeted resource group Default


Select a region (or press enter to skip):
1. au-syd
2. jp-tok
3. eu-de
4. eu-gb
5. us-south
6. us-east
Enter a number> 5
Targeted region us-south


API endpoint:      https://cloud.ibm.com
Region:            us-south
User:              [redacted]
Account:           Brew Monster's Account ([redacted])
Resource group:    Default
CF API endpoint:
Org:
Space:
```

Deploying your first Node.js application - Prerequisites: Prepare IBM Cloud (cont.)

- After you are logged in, set up the Cloud Foundry API endpoint by running the following command:
  ```
  ibmcloud target --cf
  ```
- To check the default organization and space to which you have access, go to https://cloud.ibm.com/account/cloud-foundry.

Deploying your first Node.js application - Step 1: Understanding the sample application

- Clone the sample app by running the following command:
  `git clone https://github.com/IBM-Cloud/get-started-node`
- Go to the `get-started-node` folder, which contains the following files:

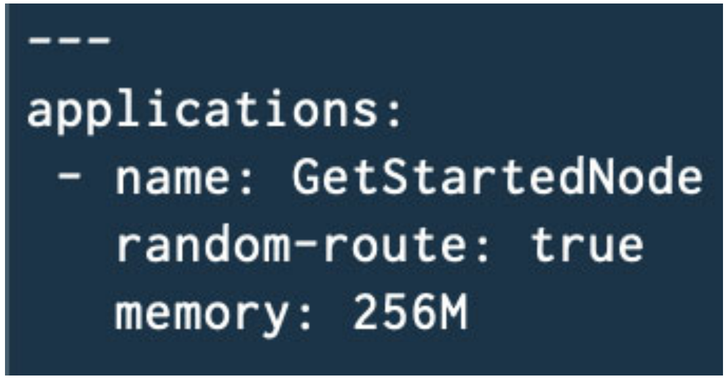


```
.
├── .cfignore
├── .git
├── .gitignore
├── Dockerfile
├── LICENSE
├── README-kubernetes.md
├── README.md
├── README_MONGO.md
├── kubernetes
├── manifest.yml
├── package-lock.json
├── package.json
├── server.js
├── vcap-local.MongoDBexample.json
├── vcap-local.example.json
└── views
```

🟩 Kubernetes/Docker	⬜ Cloud Foundry
🟥 Application specific	🟦 Documentation
🟫 Version control	

Deploying your first Node.js application - Step 1: Understanding the sample application (cont.)

- To deploy Cloud Foundry apps on IBM Cloud, you need a `manifest.yml` file.
- The sample application already includes an example of this manifest file:

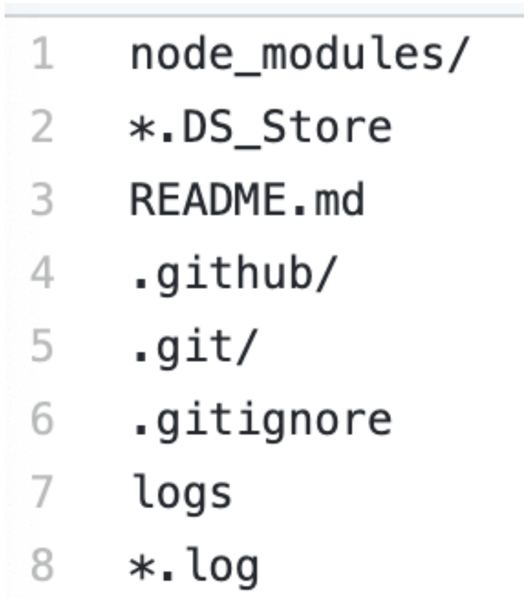


```
---
applications:
 - name: GetStartedNode
   random-route: true
   memory: 256M
```

- This manifest file lists the deployment configurations, including the name of the app, how much memory the instance should have, and the routing.

Deploying your first Node.js application - Step 1: Understanding the sample application (cont.)

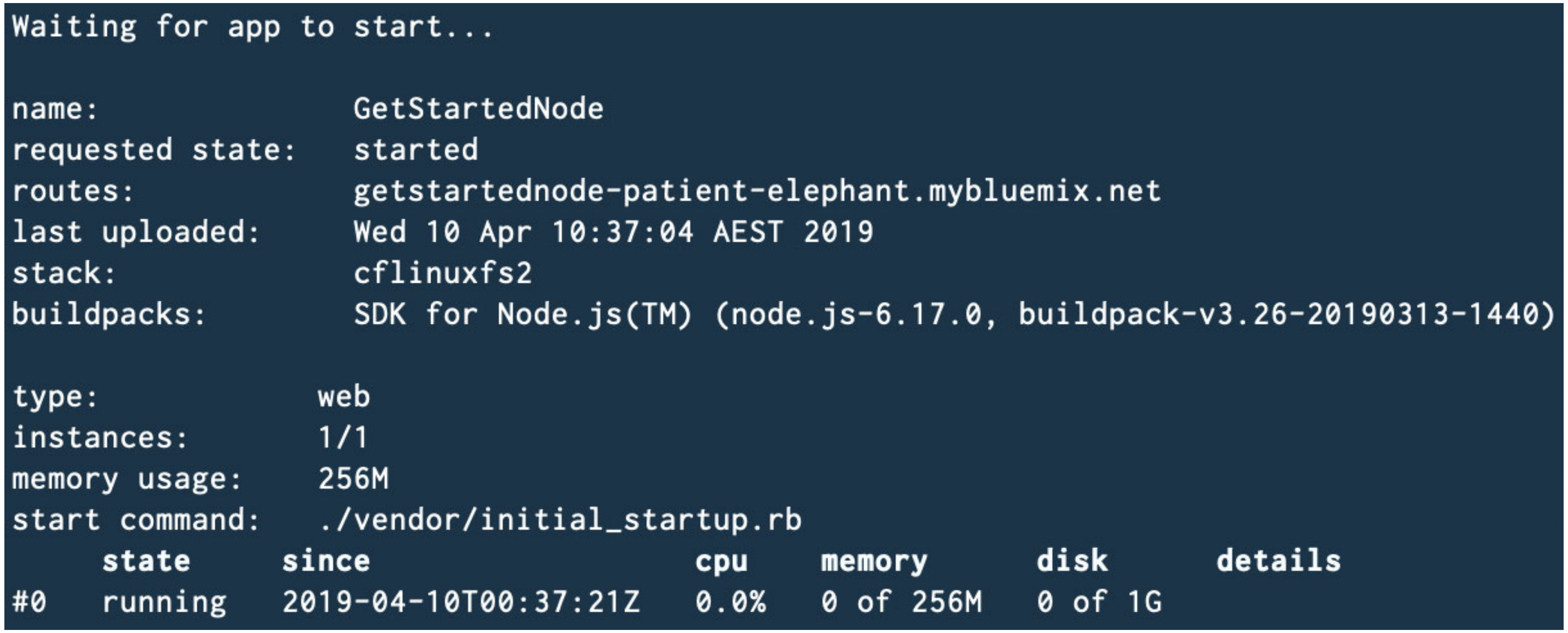
- To prevent uploading non-essential files and folders as you deploy your application, create an ignore file.
- The `.cfignore` file prevents **`cf push`** from uploading all the files and folders that are listed in the file:



```
node_modules/
*.DS_Store
README.md
.github/
.git/
.gitignore
logs
*.log
```

Deploying your first Node.js application - Step 2: Deploying the sample app

- Open the sample app folder and run **`ibmcloud cf push`**.

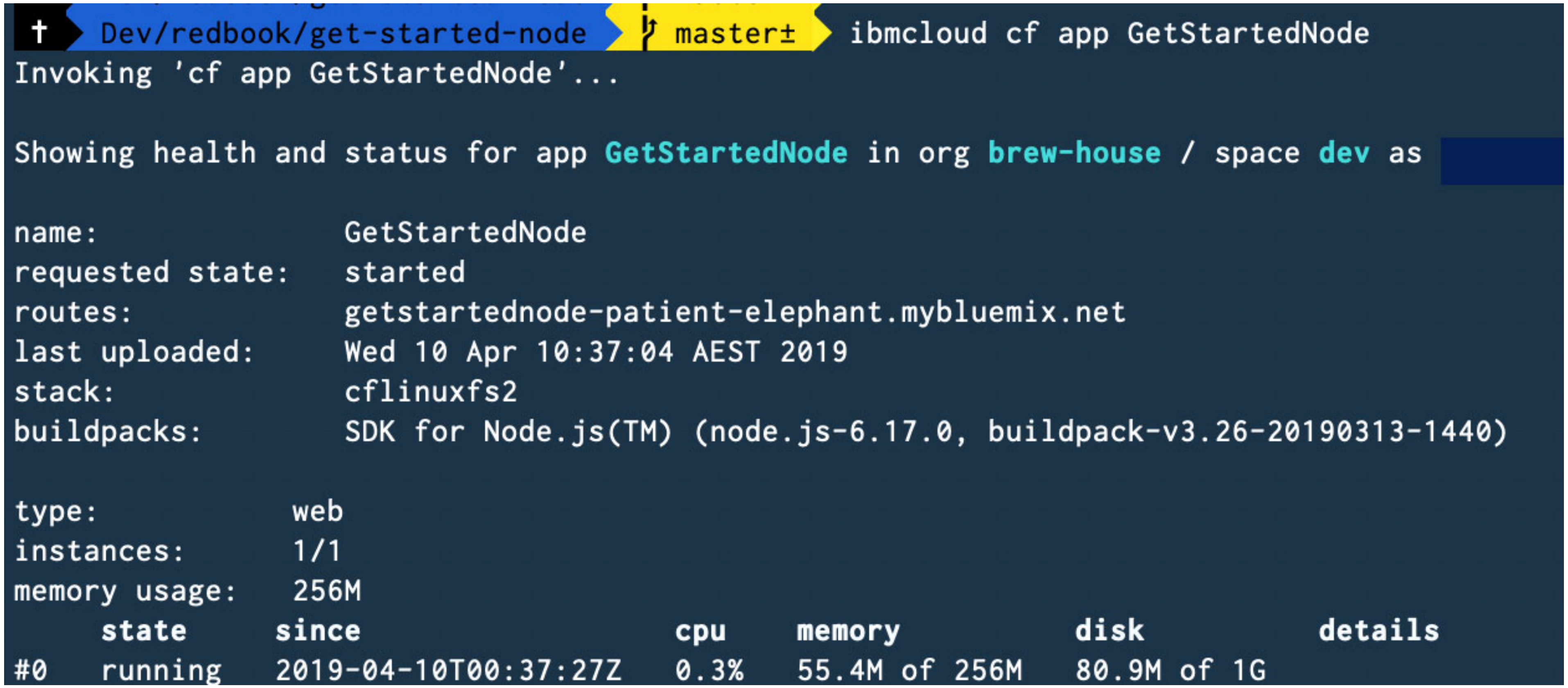


```
Waiting for app to start...

name:              GetStartedNode
requested state:   started
routes:            getstartednode-patient-elephant.mybluemix.net
last uploaded:     Wed 10 Apr 10:37:04 AEST 2019
stack:             cflinuxfs2
buildpacks:        SDK for Node.js(TM) (node.js-6.17.0, buildpack-v3.26-20190313-1440)

type:            web
instances:       1/1
memory usage:    256M
start command:   ./vendor/initial_startup.rb
     state     since                  cpu    memory       disk       details
#0   running   2019-04-10T00:37:21Z   0.0%   0 of 256M    0 of 1G
```

- Your application, which is named `GetStartedNode`, is running at a random route.

Deploying your first Node.js application - Step 3: Checking whether your app is running

- You can view the details of your app by running the following command:
  `ibmcloud cf app GetStartedNode`



```
✝ Dev/redbook/get-started-node  master± ibmcloud cf app GetStartedNode
Invoking 'cf app GetStartedNode'...

Showing health and status for app GetStartedNode in org brew-house / space dev as [redacted]

name:              GetStartedNode
requested state:   started
routes:            getstartednode-patient-elephant.mybluemix.net
last uploaded:     Wed 10 Apr 10:37:04 AEST 2019
stack:             cflinuxfs2
buildpacks:        SDK for Node.js(TM) (node.js-6.17.0, buildpack-v3.26-20190313-1440)

type:           web
instances:      1/1
memory usage:   256M
     state     since                  cpu    memory         disk          details
#0   running   2019-04-10T00:37:27Z   0.3%   55.4M of 256M  80.9M of 1G
```

Exploring your deployed application

To explore how your application directories are structured after they are deployed, use SSH to the deployed application by running the following command:

```
ibmcloud cf ssh GetStartedNode
```

Organizations and spaces

Topics

Organizations and spaces

- Cloud Foundry is a virtualized layer (containers) on top of virtual machines (VMs).
- Developers do not have direct access to the VM or machines to which they are deploying. When you push your application, Cloud Foundry provisions a logical partition of resources for it.
- Cloud Foundry uses logical boundaries to allocate resources. These are known as *Orgs* (organizations) and *Spaces*.
- These boundaries provide:
  - Separation between Cloud Foundry resources
  - Separation between teams
  - Isolation of development, test, staging, and production environments
- A developer can belong to multiple orgs and spaces.

Orgs

- *Org* is a level of abstraction to manage resources, such as service availability, quota plans, applications and custom domains for multiple users.
- With a free IBM Cloud account (Lite), you are entitled to a maximum of one org.
- When working within a team or a company, a logical mapping (of an org) might be to your business unit, an application, or to your team.
- Collaborators in an org share a resource quota plan, applications, services availability, and custom domains.

Spaces

- A *space* provides a shared location for multiple users to deploy multiple applications.
- Every space belongs to one org. Each org can have multiple spaces.
- Any developer in a space can access and edit the configurations of an application.
- Environment variables (memory per app, routes, number of instances, and app-specific variables) are contained within a space.
- Example: You can have multiple spaces, each mapped to an environment, such as dev, test, user-acceptance testing (UAT), and production environments.

Buildpacks

Topics

- Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- Resiliency
- Logging and debugging
- Domains and routes
- Binding external services
- Next steps

Introduction to buildpacks

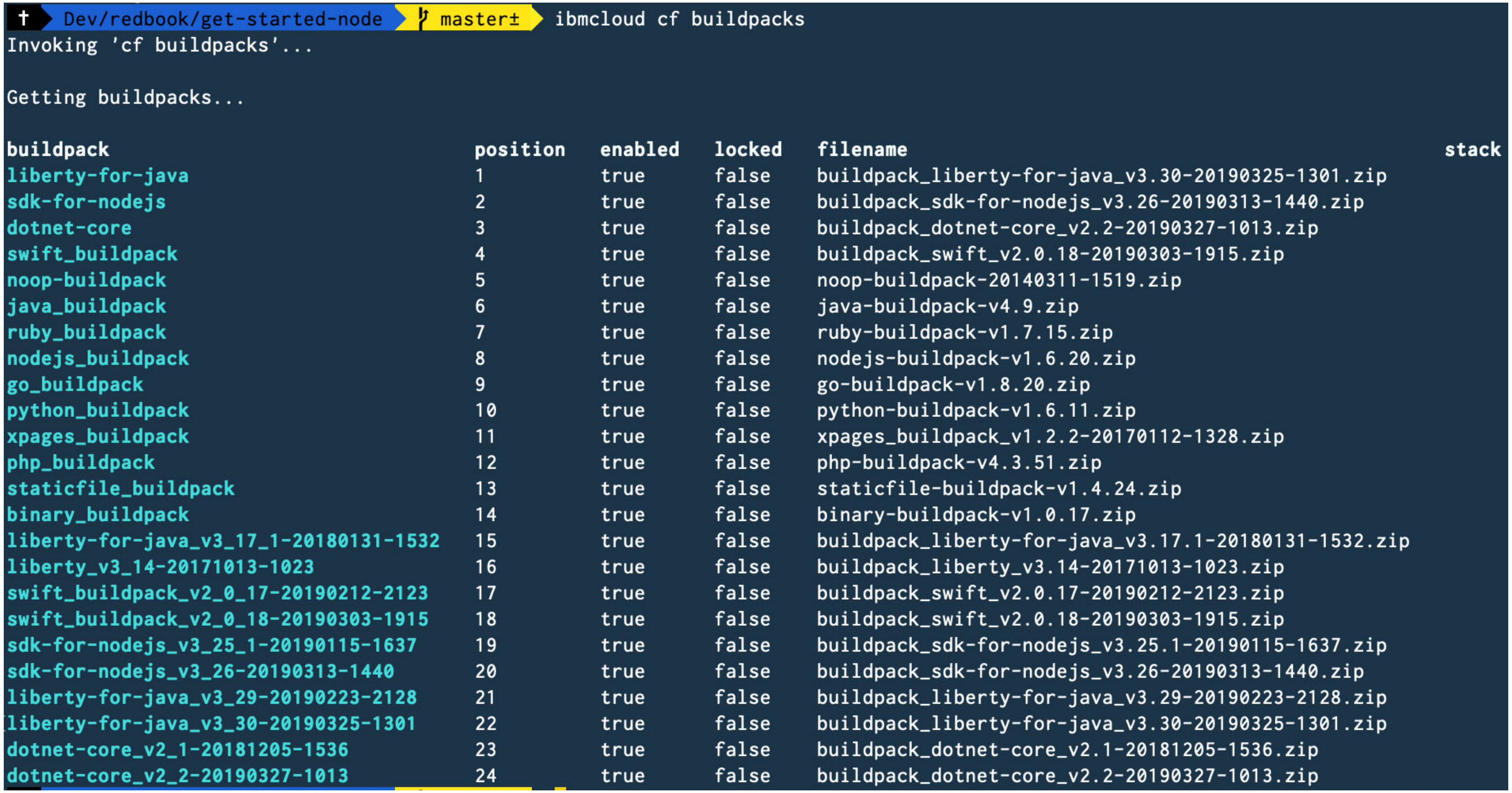
- A *buildpack* is a template and tools that help you resolve your runtime dependencies.
- For the sample application, Cloud Foundry auto-detects **`package.json`** and uses a Node.js buildpack.
- You can force Cloud Foundry to use the Node.js buildpack either by:
  - Specifying the `language` key in `manifest.yml`.
  - Pushing the application by running the following command:

    ```
    ibmcloud cf push -b BUILDPACK_NAME
    ```

IBM Cloud provided buildpacks

To get a list of the buildpacks that are available on IBM Cloud, run the following command:

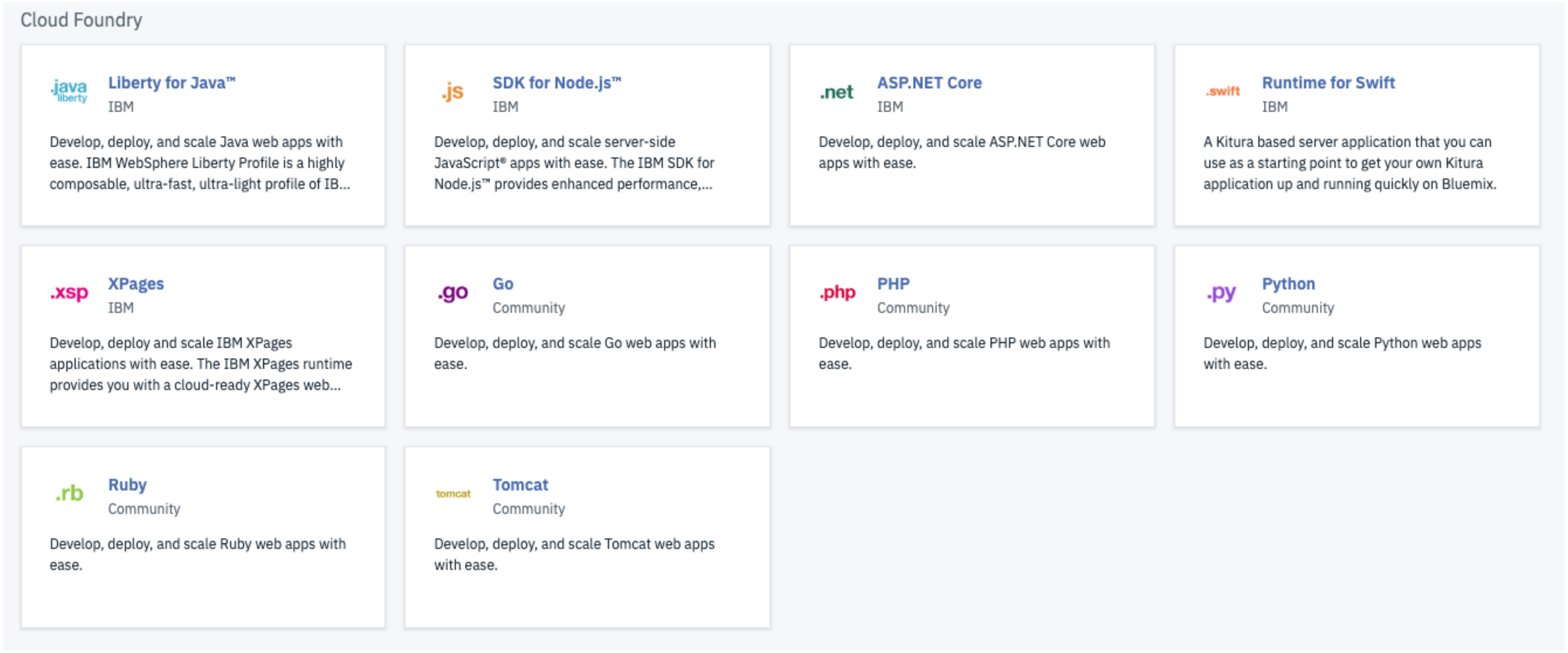
```
ibmcloud cf buildpacks
```



```
† Dev/redbook/get-started-node  master±  ibmcloud cf buildpacks
Invoking 'cf buildpacks'...

Getting buildpacks...

buildpack                             position   enabled   locked   filename                                                stack
liberty-for-java                      1          true      false    buildpack_liberty-for-java_v3.30-20190325-1301.zip
sdk-for-nodejs                        2          true      false    buildpack_sdk-for-nodejs_v3.26-20190313-1440.zip
dotnet-core                           3          true      false    buildpack_dotnet-core_v2.2-20190327-1013.zip
swift_buildpack                       4          true      false    buildpack_swift_v2.0.18-20190303-1915.zip
noop-buildpack                        5          true      false    noop-buildpack-20140311-1519.zip
java_buildpack                        6          true      false    java-buildpack-v4.9.zip
ruby_buildpack                        7          true      false    ruby-buildpack-v1.7.15.zip
nodejs_buildpack                      8          true      false    nodejs-buildpack-v1.6.20.zip
go_buildpack                          9          true      false    go-buildpack-v1.8.20.zip
python_buildpack                      10         true      false    python-buildpack-v1.6.11.zip
xpages_buildpack                      11         true      false    xpages_buildpack_v1.2.2-20170112-1328.zip
php_buildpack                         12         true      false    php-buildpack-v4.3.51.zip
staticfile_buildpack                  13         true      false    staticfile-buildpack-v1.4.24.zip
binary_buildpack                      14         true      false    binary-buildpack-v1.0.17.zip
liberty-for-java_v3_17_1-20180131-1532 15        true      false    buildpack_liberty-for-java_v3.17.1-20180131-1532.zip
liberty_v3_14-20171013-1023           16         true      false    buildpack_liberty_v3.14-20171013-1023.zip
swift_buildpack_v2_0_17-20190212-2123 17         true      false    buildpack_swift_v2.0.17-20190212-2123.zip
swift_buildpack_v2_0_18-20190303-1915 18         true      false    buildpack_swift_v2.0.18-20190303-1915.zip
sdk-for-nodejs_v3_25_1-20190115-1637  19         true      false    buildpack_sdk-for-nodejs_v3.25.1-20190115-1637.zip
sdk-for-nodejs_v3_26-20190313-1440    20         true      false    buildpack_sdk-for-nodejs_v3.26-20190313-1440.zip
liberty-for-java_v3_29-20190223-2128  21         true      false    buildpack_liberty-for-java_v3.29-20190223-2128.zip
liberty-for-java_v3_30-20190325-1301  22         true      false    buildpack_liberty-for-java_v3.30-20190325-1301.zip
dotnet-core_v2_1-20181205-1536        23         true      false    buildpack_dotnet-core_v2.1-20181205-1536.zip
dotnet-core_v2_2-20190327-1013        24         true      false    buildpack_dotnet-core_v2.2-20190327-1013.zip
```

IBM Cloud catalog view of buildpacks

IBM Cloud provides a UI catalog of the previously mentioned buildpacks.



Cloud Foundry

Liberty for Java™
IBM
Develop, deploy, and scale Java web apps with ease. IBM WebSphere Liberty Profile is a highly composable, ultra-fast, ultra-light profile of IB...

SDK for Node.js™
IBM
Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance,...

ASP.NET Core
IBM
Develop, deploy, and scale ASP.NET Core web apps with ease.

Runtime for Swift
IBM
A Kitura based server application that you can use as a starting point to get your own Kitura application up and running quickly on Bluemix.

XPages
IBM
Develop, deploy and scale IBM XPages applications with ease. The IBM XPages runtime provides you with a cloud-ready XPages web...

Go
Community
Develop, deploy, and scale Go web apps with ease.

PHP
Community
Develop, deploy, and scale PHP web apps with ease.

Python
Community
Develop, deploy, and scale Python web apps with ease.

Ruby
Community
Develop, deploy, and scale Ruby web apps with ease.

Tomcat
Community
Develop, deploy, and scale Tomcat web apps with ease.

Using custom buildpacks

- Aside from IBM buildpacks, you can also use community buildpacks and even create your own buildpacks.
- To use a custom buildpack, you can either:
  - Add the `buildpack` key into the application's `manifest.yml` file

```
---
applications:
- name:
  memory: 128M
  buildpack: GIT_BUILDPACK_URL
```

  - Specify a Git URL to the buildpack, for example:

```
ibmcloud cf push APP_NAME -b https://github.com/cloudfoundry/java-
buildpack.git
```

Resiliency

Topics

- Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- ▶ Resiliency
- Logging and debugging
- Domains and routes
- Binding external services
- Next steps

Making your app resilient

- Resiliency is about ensuring that the actual system state (number of running applications) matches the wanted state always.
- To configure the number of instances that you want in `manifest.yml`, specify the number of instances in the `instances` key.
- In the event of a failure (failed system process, unresponsive containers, and so on), Cloud Foundry kills or re-creates missing instances to match the wanted state.

```
applications:
- name: myCustomApp
  memory: 128M
  instances: 2
```

Making your app resilient by using IBM Cloud CLI

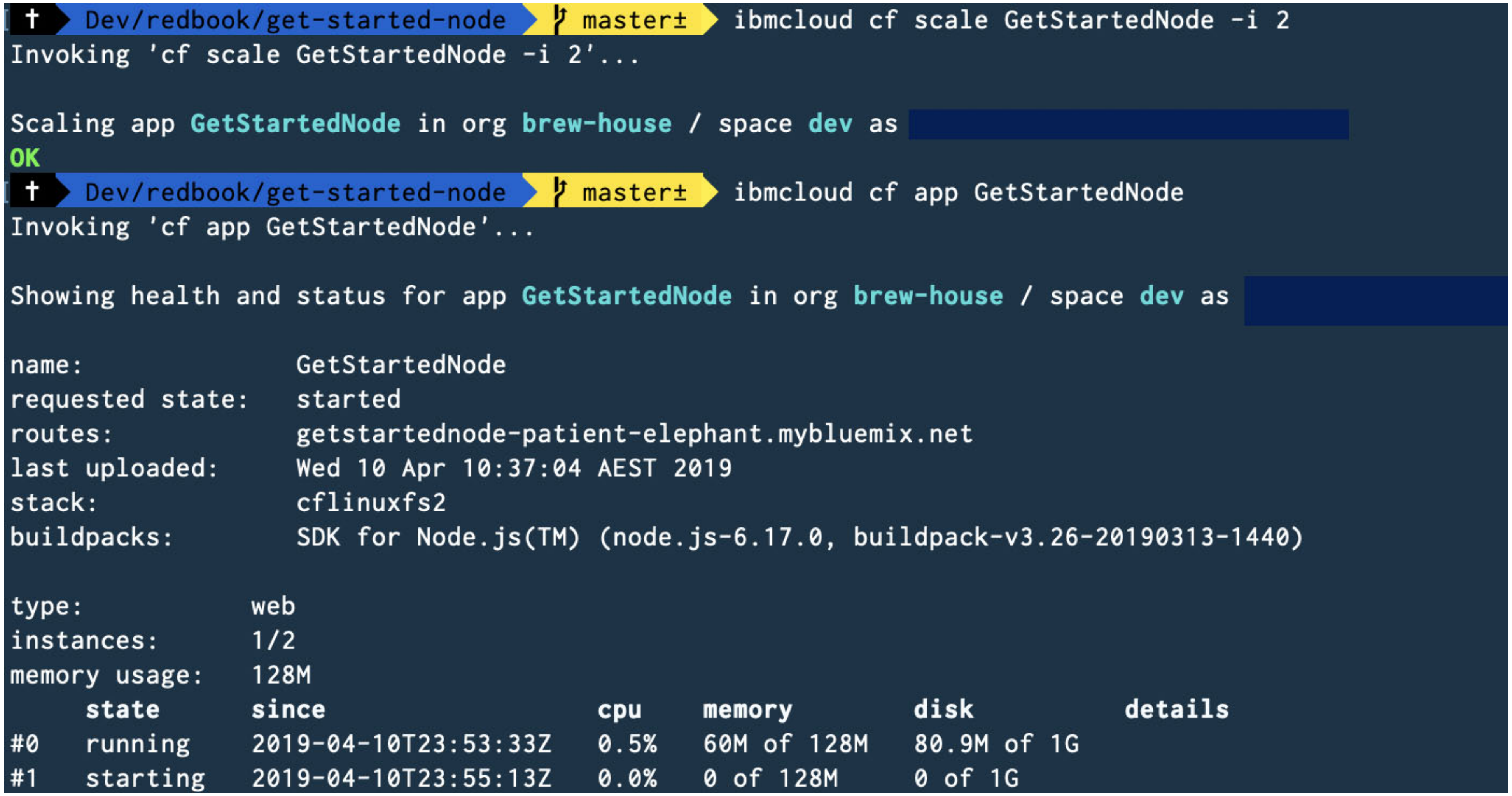
- Aside from specifying the number of instances in `manifest.yml`, you can also use the IBM Cloud CLI to scale your application by running the following command:

```
ibmcloud cf scale APP_NAME -i NUMBER_OF_INSTANCES
```

- However, it is recommended that you specify the number of wanted instances in your `manifest.yml` file to ensure that your configurations are stored as code.
- When you push your application, the value in your manifest file overrides any custom configuration.

Making your app resilient by using IBM Cloud CLI (cont.)

- Example: `ibmcloud cf scale GetStartedNode -i 2`



```
[ † Dev/redbook/get-started-node  master± ibmcloud cf scale GetStartedNode -i 2
Invoking 'cf scale GetStartedNode -i 2'...

Scaling app GetStartedNode in org brew-house / space dev as [redacted]
OK
[ † Dev/redbook/get-started-node  master± ibmcloud cf app GetStartedNode
Invoking 'cf app GetStartedNode'...

Showing health and status for app GetStartedNode in org brew-house / space dev as [redacted]

name:              GetStartedNode
requested state:   started
routes:            getstartednode-patient-elephant.mybluemix.net
last uploaded:     Wed 10 Apr 10:37:04 AEST 2019
stack:             cflinuxfs2
buildpacks:        SDK for Node.js(TM) (node.js-6.17.0, buildpack-v3.26-20190313-1440)

type:           web
instances:      1/2
memory usage:   128M
     state      since                  cpu    memory        disk         details
#0   running    2019-04-10T23:53:33Z   0.5%   60M of 128M   80.9M of 1G
#1   starting   2019-04-10T23:55:13Z   0.0%   0 of 128M     0 of 1G
```

Logging and debugging

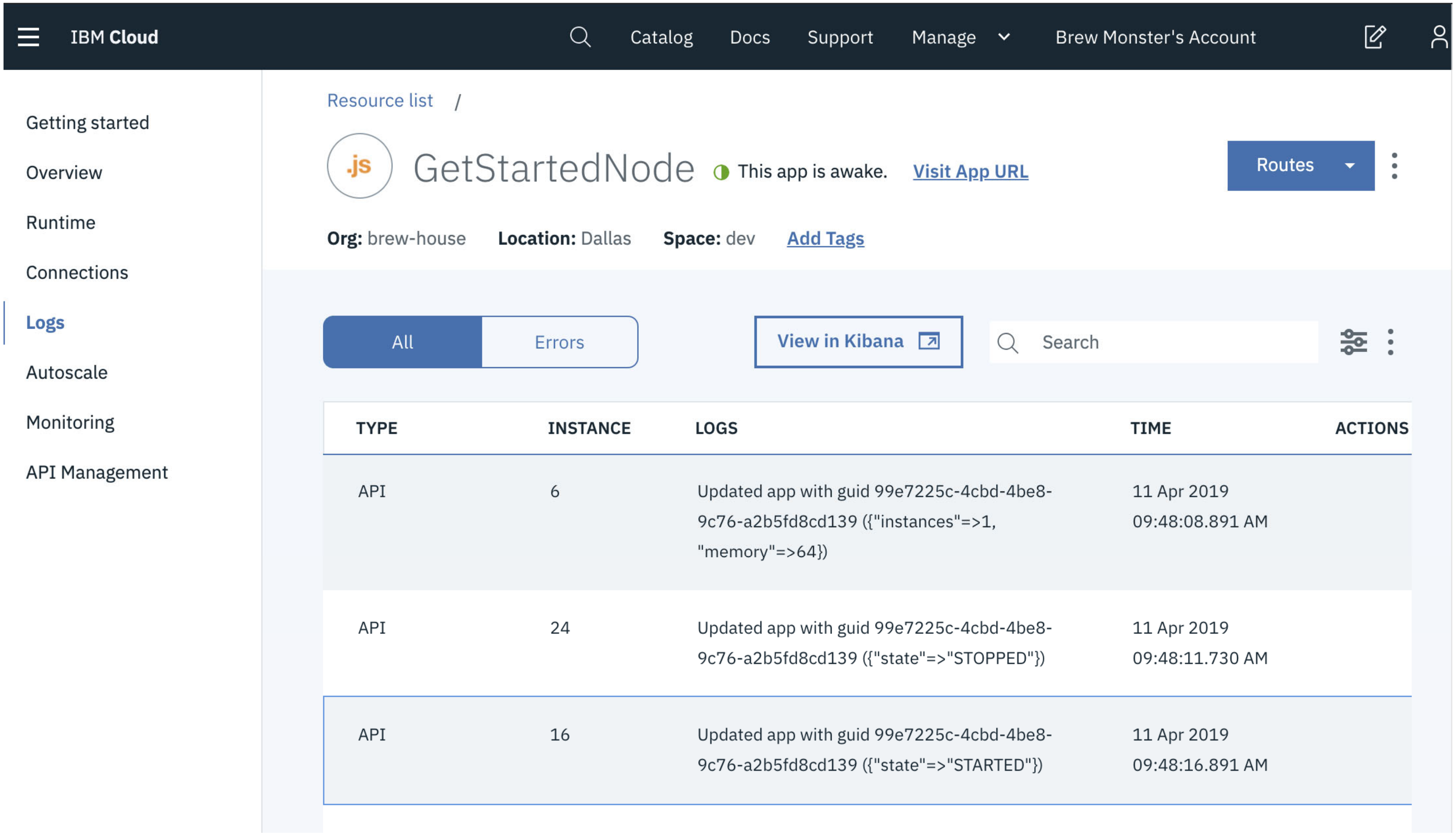
Topics

- Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- Resiliency
- ▶ Logging and debugging
- Domains and routes
- Binding external services
- Next steps

Debugging your application’s deployment

- The Cloud Foundry platform provides log aggregations.
- To view events and logs for the deployed application, run the following commands:
  - `ibmcloud cf events GetStartedNode`
  - `ibmcloud cf logs GetStartedNode`
- Additionally, you can use the IBM Cloud UI to look at the logs of your application.

Viewing logs from the IBM Cloud dashboard



Org: brew-house Location: Dallas Space: dev Add Tags

TYPE	INSTANCE	LOGS	TIME	ACTIONS
API	6	Updated app with guid 99e7225c-4cbd-4be8-9c76-a2b5fd8cd139 ({"instances"=>1, "memory"=>64})	11 Apr 2019 09:48:08.891 AM	
API	24	Updated app with guid 99e7225c-4cbd-4be8-9c76-a2b5fd8cd139 ({"state"=>"STOPPED"})	11 Apr 2019 09:48:11.730 AM	
API	16	Updated app with guid 99e7225c-4cbd-4be8-9c76-a2b5fd8cd139 ({"state"=>"STARTED"})	11 Apr 2019 09:48:16.891 AM	

Domains and routes

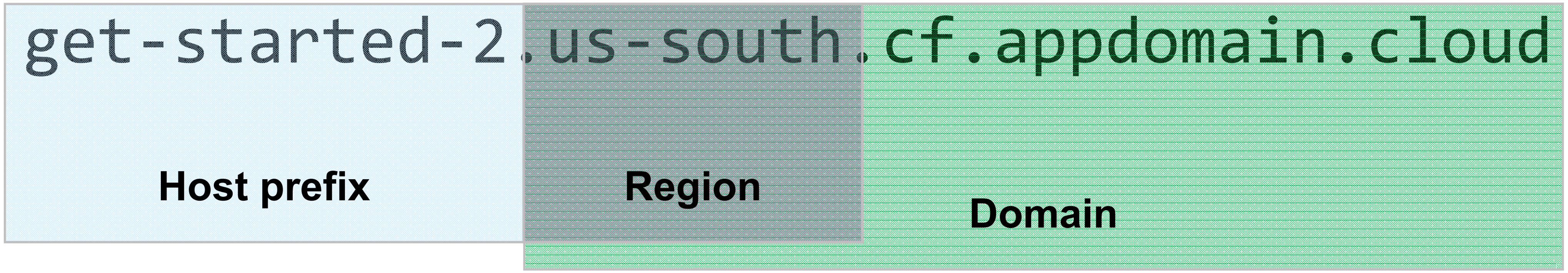
Topics

- Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- Resiliency
- Logging and debugging
- ▶ Domains and routes
- Binding external services
- Next steps

Domains and routes

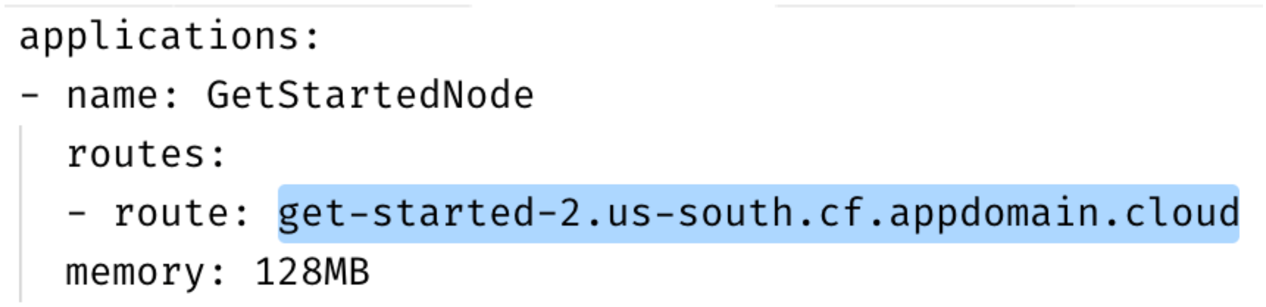
- Domains and routes enable traffic from the internet to flow to and from your application.
- A route (URL) is composed of a host prefix and a domain.
- IBM Cloud provides domains for each region. The developer specifies a *unique* host prefix for an application.
- Each application can have multiple routes.
- One route can serve multiple applications by using path routing.

Example of a route



Setting up routes

- In the sample manifest, `random-route` is set to `true` to avoid duplicated host prefixes.
- To specify a route to an application, you can either:
  - Configure the route by using `manifest.yml`.



```
applications:
- name: GetStartedNode
  routes:
  - route: get-started-2.us-south.cf.appdomain.cloud
  memory: 128MB
```

  - Use the CLI with a unique host by running the following command:

```
ibmcloud app route-map APP_NAME DOMAIN -n HOST
```

Binding external services

Topics

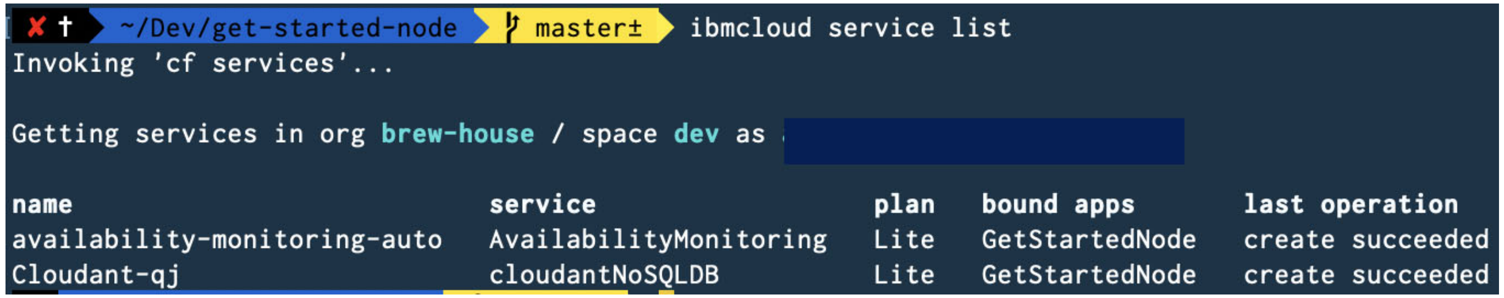
- Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- Resiliency
- Logging and debugging
- Domains and routes
- ▶ Binding external services
- Next steps

External services

- On Cloud Foundry, databases, file systems, messaging services, and any external systems with which your application interacts, are called *services*.
- These services enable you to do the following actions:
  - Horizontally scale.
  - Use existing marketplace services.
  - Reduce complexity and routing costs.
- For a list of services that are available on IBM Cloud, run the following command:
  - `ibmcloud service offerings`
  - Use the web UI to create services.

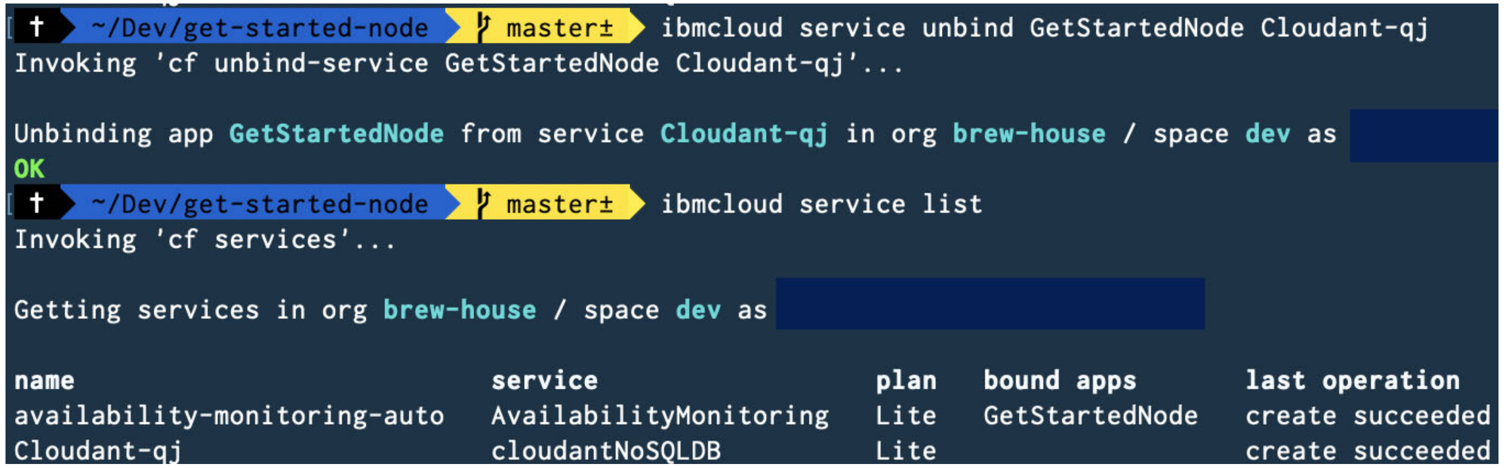
Example of bound services

- As an example, GetStartedNode app is bound to two services: availability monitoring and IBM Cloudant.
- To view a list of services to which an app is bound, run the following command:
  `ibmcloud service list`



```
 ✘ † ~/Dev/get-started-node  master± ibmcloud service list
Invoking 'cf services'...

Getting services in org brew-house / space dev as [redacted]

name                          service                 plan   bound apps       last operation
availability-monitoring-auto  AvailabilityMonitoring  Lite   GetStartedNode   create succeeded
Cloudant-qj                   cloudantNoSQLDB         Lite   GetStartedNode   create succeeded
```

- To unbind a service, run the following command:
  `ibmcloud service unbind APP_NAME SERVICE_NAME`

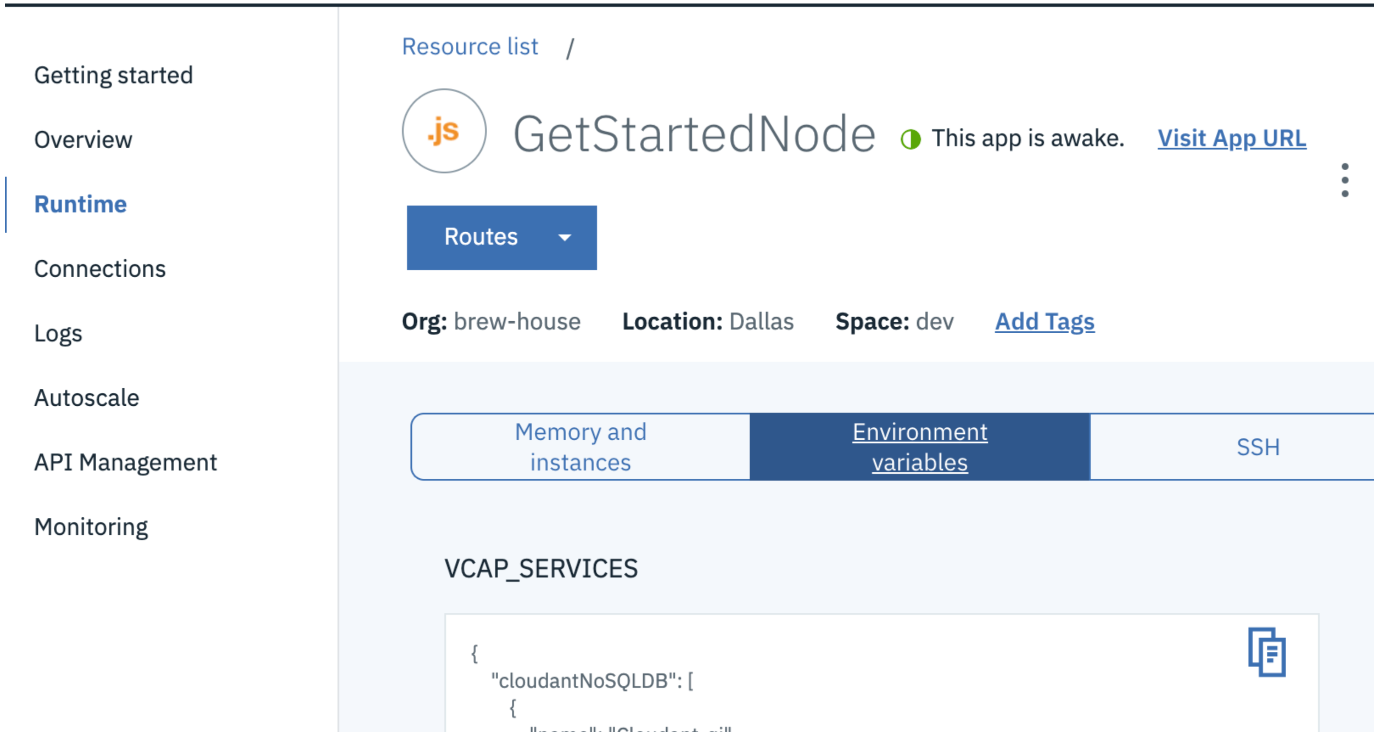


```
 † ~/Dev/get-started-node  master± ibmcloud service unbind GetStartedNode Cloudant-qj
Invoking 'cf unbind-service GetStartedNode Cloudant-qj'...

Unbinding app GetStartedNode from service Cloudant-qj in org brew-house / space dev as [redacted]
OK
 † ~/Dev/get-started-node  master± ibmcloud service list
Invoking 'cf services'...

Getting services in org brew-house / space dev as [redacted]

name                          service                 plan   bound apps       last operation
availability-monitoring-auto  AvailabilityMonitoring  Lite   GetStartedNode   create succeeded
Cloudant-qj                   cloudantNoSQLDB         Lite                    create succeeded
```

Bound services in environment variables

- After a service is bound to an application, IBM Cloud restarts the application and provides the credentials of the service to the app by using `VCAP_SERVICES` environment variables.
- You can view this variable by clicking **Runtime** on the web UI sidebar and clicking **Environment variables**, or by running the following command:

```
ibmcloud cf env APP_NAME
```



Org: brew-house Location: Dallas Space: dev Add Tags

VCAP_SERVICES

```
{
  "cloudantNoSQLDB": [
    {
```

Next steps

Topics

- Introduction to Cloud Foundry
- Deploying Cloud Foundry applications with IBM Cloud CLI
- Organizations and spaces
- Buildpacks
- Resiliency
- Logging and debugging
- Domains and routes
- Binding external services
- ▶ Next steps

Further reading

- In this unit, you learned the basic concepts and tenets of Cloud Foundry.
- To become better versed with this technology, see the following resources:
  - All attributes of `manifest.yml`:
    https://docs.cloudfoundry.org/devguide/deploy-apps/manifest-attributes.html
  - App container lifecycle:
    https://docs.cloudfoundry.org/devguide/deploy-apps/app-lifecycle.html
  - How to deploy Docker images onto Cloud Foundry:
    https://docs.cloudfoundry.org/devguide/deploy-apps/push-docker.html
  - Security and credential management with CredHub:
    https://docs.cloudfoundry.org/credhub/index.html
  - BOSH:
    https://www.cloudfoundry.org/bosh/

Unit summary

- Explain how to manage your IBM Cloud account with the IBM Cloud CLI.
- Describe how to create a Node.js application that runs on IBM Cloud.
- Deploy an application from a local workstation by using the IBM Cloud CLI.
- Describe the role of Node.js for server-side scripting.
- Deploy an application by using IBM Cloud App Service (Web Apps).

Exercise 1: Getting started with Cloud Foundry apps on IBM Cloud

Exercise objectives

- This exercise describes how you can deploy a web application (app) without downloading or configuring a runtime environment or framework or setting up a server. This exercise also covers how to test and run the app when it is deployed.
- After completing this exercise, you should be able to:
  - Log in to IBM Cloud from a browser.
  - Create an IBM Cloud application by using one of the available run times.
  - Install the IBM Cloud command-line interface (CLI).
  - Sign on to IBM Cloud from the CLI.
  - Deploy an application from a local workstation by using the IBM Cloud CLI.
  - Test the application with its endpoint after the application is deployed and started.