1. Do you understand Cloud Foundry concepts like spaces, organizations, routes,

services, domains, users, quotas?

<https://docs.cloudfoundry.org/devguide/deploy-apps/routes-domains.html>   
<https://docs.cloudfoundry.org/concepts/roles.html>   
<https://docs.cloudfoundry.org/#read-the-docs>

<https://docs.cloudfoundry.org/devguide/services/>

<http://it.psionline.com/exam-faqs/pivotal-faq>

[https://www.examslocal.com](https://www.examslocal.com/)

Cloud Foundry Overview

Q 5. What is the difference between a public, private and hybrid cloud?

Cloud Deployment Models

The cloud deployment models summarized below are the following:

* **Private Cloud:**the cloud services used by a single organization, which are not exposed to the public. A private cloud resides inside the organization and must be behind a firewall, so only the organization has access to it and can manage it.
* **Public Cloud:** the cloud services are exposed to the public and can be used by anyone. Virtualization is typically used to build the cloud services that are offered to the public. An example of a public cloud is Amazon Web Services (AWS).
* **Hybrid Cloud:** the cloud services can be distributed among public and private clouds, where sensitive applications are kept inside the organization’s network (by using a private cloud), whereas other services can be hosted outside the organization’s network (by using a public cloud). Users can them interchangeably use private as well as public cloud services in every day operations.

The biggest differences between public, private and hybrid cloud are described in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Difference** | **Private** | **Public** | **Hybrid** |
| **Tenancy** | Single tenancy: there’s only the data of a single organization stored in the cloud. | Multi-tenancy: the data of multiple organizations in stored in a shared environment. | The data stored in the public cloud is usually multi-tenant, which means the data from multiple organizations is stored in a shared environment. The data stored in private cloud is kept private by the organization. |
| **Exposed to the Public** | No: only the organization itself can use the private cloud services. | Yes: anyone can use the public cloud services. | The services running on a private cloud can be accessed only the organization’s users, while the services running on public cloud can be accessed by anyone. |
| **Data Center Location** | Inside the organization’s network. | Anywhere on the Internet where the cloud service provider’s services are located. | Inside the organization’s network for private cloud services as well as anywhere on the Internet for public cloud services. |
| **Cloud Service Management** | The organization must have their own administrators managing their private cloud services. | The cloud service provider manages the services, where the organization merely uses them. | The organization itself must manage the private cloud, while the public cloud is managed by the CSP. |
| **Hardware Components** | Must be provided by the organization itself, which has to buy physical servers to build the private cloud on. | The CSP provides all the hardware and ensures it’s working at all times. | The organization must provide hardware for the private cloud, while the hardware of CSP is used for public cloud services. |
| **Expenses** | Can be quite expensive, since the hardware, applications and network have to be provided and managed by the organization itself. | The CSP has to provide the hardware, set-up the application and provide the network accessibility according to the SLA. | The private cloud services must be provided by the organization, including the hardware, applications and network, while the CSP manages the public cloud services. |

As you can see, the hybrid cloud is a combination of private, as well as public cloud, used together by the same organization to pull the best features from each.

Which one should you choose?

It’s important to keep in mind when deciding whether to build a private or public cloud, to properly weigh the differences against each other. In most cases they can be thought of as advantages or disadvantages, depending on the usage required. If we’d like to store our backup data somewhere in the cloud, it’s important to determine the sensitivity of said data. For example, if we are storing confidential information such as credit card information or medical records we absolutely must store that data in a private cloud but when it comes to non-sensitive info, we can store it in a public cloud if it keeps costs down considerably.

Then, there’s always a choice whether to integrate public or private cloud into our everyday operations. Again, there are advantages and disadvantages that need to be taken into consideration. Whether to compromise the security and keep expenses down or pay a little extra for additional layer of security is a choice you’ll sooner or later have to make; but you should always consider what’s best for your business and move forward from there.

Q6.What infrastructures does Cloud Foundry run on?

## The Industry-Standard Cloud Platform

Cloud platforms let anyone deploy network apps or services and make them available to the world in a few minutes. When an app becomes popular, the cloud easily scales it to handle more traffic, replacing with a few keystrokes the build-out and migration efforts that once took months. Cloud platforms represent the next step in the evolution of IT, enabling you to focus exclusively on your applications and data without worrying about underlying infrastructure.

Not all cloud platforms are created equal. Some have limited language and framework support, lack key app services, or restrict deployment to a single cloud. Cloud Foundry (CF) has become the industry standard. It is an [open source](https://github.com/cloudfoundry) platform that you can deploy to run your apps on your own computing infrastructure, or deploy on an IaaS like AWS, vSphere, or OpenStack. You can also use a PaaS deployed by a commercial [CF cloud provider](https://www.cloudfoundry.org/learn/certified-providers/). A broad [community](https://www.cloudfoundry.org/community/) contributes to and supports Cloud Foundry. The platform’s openness and extensibility prevent its users from being locked into a single framework, set of app services, or cloud.

Cloud Foundry is ideal for anyone interested in removing the cost and complexity of configuring infrastructure for their apps. Developers can deploy their apps to Cloud Foundry using their existing tools and with zero modification to their code.

## How Cloud Foundry Works

To flexibly serve and scale apps online, Cloud Foundry has subsystems that perform specialized functions. Here’s how some of these main subsystems work.

### How the Cloud Balances Its Load

Clouds balance their processing loads over multiple machines, optimizing for efficiency and resilience against point failure. A Cloud Foundry installation accomplishes this at three levels:

1. [BOSH](http://bosh.io/) creates and deploys virtual machines (VMs) on top of a physical computing infrastructure, and deploys and runs Cloud Foundry on top of this cloud. To configure the deployment, BOSH follows a manifest document.
2. The CF [Cloud Controller](https://docs.cloudfoundry.org/concepts/architecture/cloud-controller.html) runs the apps and other processes on the cloud’s VMs, balancing demand and managing app lifecycles.
3. The [router](https://docs.cloudfoundry.org/concepts/architecture/router.html) routes incoming traffic from the world to the VMs that are running the apps that the traffic demands, usually working with a customer-provided load balancer.

How Apps Run Anywhere

Cloud Foundry designates two types of VMs: the component VMs that constitute the platform’s infrastructure, and the host VMs that host apps for the outside world. Within CF, the Diego system distributes the hosted app load over all of the host VMs, and keeps it running and balanced through demand surges, outages, or other changes. Diego accomplishes this through an auction algorithm.

To meet demand, multiple host VMs run duplicate instances of the same app. This means that apps must be portable. Cloud Foundry distributes app source code to VMs with everything the VMs need to compile and run the apps locally. This includes the OS [stack](https://docs.cloudfoundry.org/devguide/deploy-apps/stacks.html) that the app runs on, and a [buildpack](https://docs.cloudfoundry.org/buildpacks/) containing all languages, libraries, and services that the app uses. Before sending an app to a VM, the Cloud Controller [stages](https://docs.cloudfoundry.org/concepts/how-applications-are-staged.html) it for delivery by combining stack, buildpack, and source code into a droplet that the VM can unpack, compile, and run. For simple, standalone apps with no dynamic pointers, the droplet can contain a pre-compiled executable instead of source code, language, and libraries.

### How CF Organizes Users and Workspaces

To organize user access to the cloud and to control resource use, a cloud operator defines [Orgs and Spaces](https://docs.cloudfoundry.org/concepts/roles.html) within an installation and assigns Roles such as admin, developer, or auditor to each user. The [User Authentication and Authorization](https://docs.cloudfoundry.org/concepts/architecture/uaa.html) (UAA) server supports access control as an [OAuth2](http://oauth.io/) service, and can store user information internally or connect to external user stores through LDAP or SAML.

Where CF Stores Resources

Cloud Foundry uses the git system on [GitHub](http://github.org/) to version-control source code, buildpacks, documentation, and other resources. Developers on the platform also use GitHub for their own apps, custom configurations, and other resources. To store large binary files, such as droplets, CF maintains an internal or external blobstore. To store and share temporary information, such as internal component states, CF uses MySQL, [Consul](https://github.com/hashicorp/consul), and [etcd](https://github.com/coreos/etcd).

### How CF Components Communicate

Cloud Foundry components communicate with each other by posting messages internally using http and https protocols, and by sending [NATS](https://docs.cloudfoundry.org/concepts/architecture/messaging-nats.html) messages to each other directly.

### How to Monitor and Analyze a CF Deployment

As the cloud operates, the Cloud Controller VM, router VM, and all VMs running apps continuously generate logs and metrics. The Loggregator system aggregates this information in a structured, usable form, the **Firehose**. You can use all of the output of the Firehose, or direct the output to specific uses, such as monitoring system internals or analyzing user behavior, by applying **nozzles**.

### Using Services with CF

Typical apps depend on free or metered [services](http://docs.cloudfoundry.org/services/overview.html) such as databases or third-party APIs. To incorporate these into an app, a developer writes a Service Broker, an API that publishes to the Cloud Controller the ability to list service offerings, provision the service, and enable apps to make calls out to it.

Q6.What is BOSH? Why is it useful?

BOSH is a project that unifies release engineering, deployment, and lifecycle management of small and large-scale cloud software. BOSH can provision and deploy software over hundreds of VMs. It also performs monitoring, failure recovery, and software updates with zero-to-minimal downtime.

While BOSH was developed to deploy Cloud Foundry PaaS, it can also be used to deploy almost any other software (Hadoop, for instance). BOSH is particularly well-suited for large distributed systems. In addition BOSH supports multiple Infrastructure as a Service (IaaS) providers like VMware vSphere, vCloud Director, Amazon Web Services EC2, and OpenStack. There is a Cloud Provider Interface (CPI) that enables users to extend BOSH to support additional IaaS providers such as Google Compute Engine and Apache CloudStack

Q7. What is staging? What does it do?-->to do later

Q8. Do you know the difference between restarting, restaging and redeploying and

application? How does each of these affect the services, environment-variables available to an application?

## Start Your Application

To start your application, run the following command from your application root directory:

$ cf push YOUR-APP

For more information about deploying applications, see the [Deploy an Application](https://docs.cloudfoundry.org/devguide/deploy-apps/deploy-app.html) topic.

Cloud Foundry determines the start command for your application from one of the three following sources:

* The -c command-line option in the Cloud Foundry Command Line Interface (cf CLI). See the following example:

$ cf push YOUR-APP -c "node YOUR-APP.js"

* The command attribute in the application manifest. See the following example:

command: node YOUR-APP.js

* The buildpack, which provides a start command appropriate for a particular type of application.

The source that Cloud Foundry uses depends on factors explained below.

### How Cloud Foundry Determines its Default Start Command

The first time you deploy an application, cf push uses the buildpack start command by default. After that, cf push defaults to whatever start command was used for the previous push.

To override these defaults, provide the -c option, or the command attribute in the manifest. When you provide start commands both at the command line and in the manifest, cf push ignores the command in the manifest.Forcing Cloud Foundry To Use the Buildpack Start Command

To force Cloud Foundry to use the buildpack start command, specify a start command of null.

You can specify a null start command in one of two ways.

* Using the -c command-line option in the cf CLI:

$ cf push YOUR-APP -c "null"

* Using the command attribute in the application manifest:

command: null

This can be helpful after you have deployed while providing a start command at the command line or the manifest. At this point, a command that you provided, rather than the buildpack start command, has become the default start command. In this situation, if you decide to deploy using the buildpack start command, the null command makes that easy.

### Start Commands When Migrating a Database

Start commands are used in special ways when you migrate a database as part of an application deployment. See the [Migrating a Database in Cloud Foundry](https://docs.cloudfoundry.org/devguide/services/migrate-db.html) topic for more information.

## Restart Your Application

To restart your application, run the following command:

$ cf restart YOUR-APP

Restarting your application stops your application and restarts it with the already compiled droplet. A droplet is a tarball that includes:

* stack
* [buildpack](https://docs.cloudfoundry.org/buildpacks/)
* application source code

The Diego [cell](https://docs.cloudfoundry.org/concepts/architecture/#diego-cell) unpacks, compiles, and runs a droplet on a container.

Restart your application to refresh the application’s environment after actions such as binding a new service to the application or setting an environment variable that only the application consumes. However, if your environment variable is consumed by the buildpack in addition to the application, then you must [restage](https://docs.cloudfoundry.org/devguide/deploy-apps/start-restart-restage.html#restage) the application for the change to take effect

Restage Your Application

To restage your application, run the following command:

$ cf restage YOUR-APP

Restaging your application stops your application and restages it, by compiling a new droplet and starting it.

Restage your application if you have changed the environment in a way that affects your staging process, such as setting an environment variable that the buildpack consumes. The staging process has access to environment variables, so the environment can affect the contents of the droplet.

Restaging your application compiles a new droplet from your application without updating your application source. If you need to update your application source, re-push your application by following the steps in the section [above](https://docs.cloudfoundry.org/devguide/deploy-apps/start-restart-restage.html#start).