**Cloud Foundry for Developers**

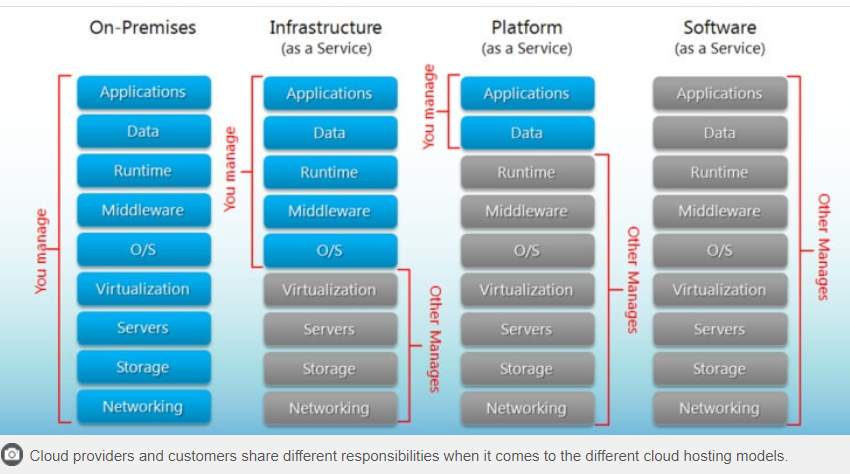
**1.Introduction**

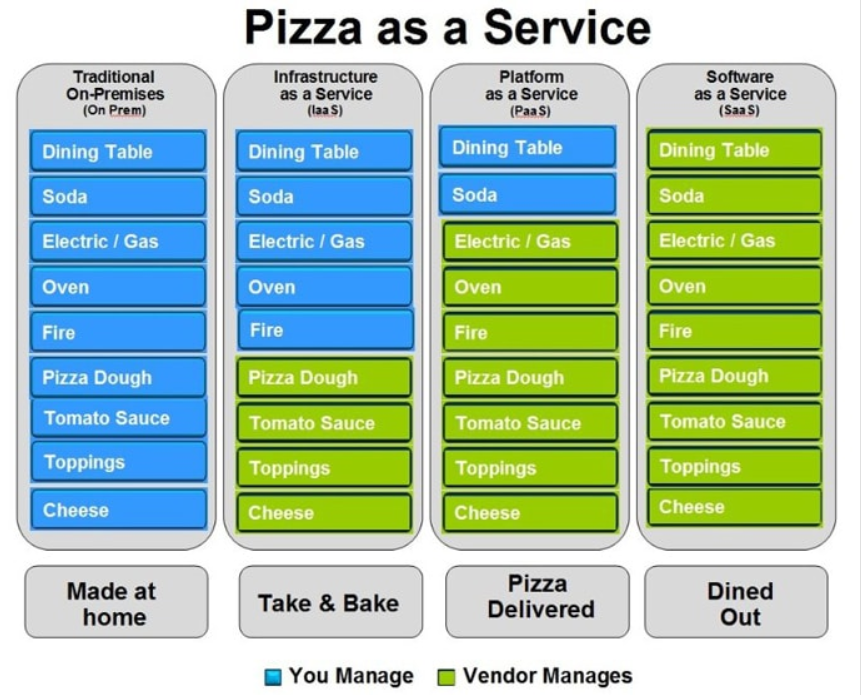
**Cloud Application Delivery Models**

IAAS

PAAS

SAAS

****

****

**Deploy an Application**

****

Nodejs.org – To install node js

<https://github.com/cloudfoundry/cli/> - To get the cloud foundry CLI

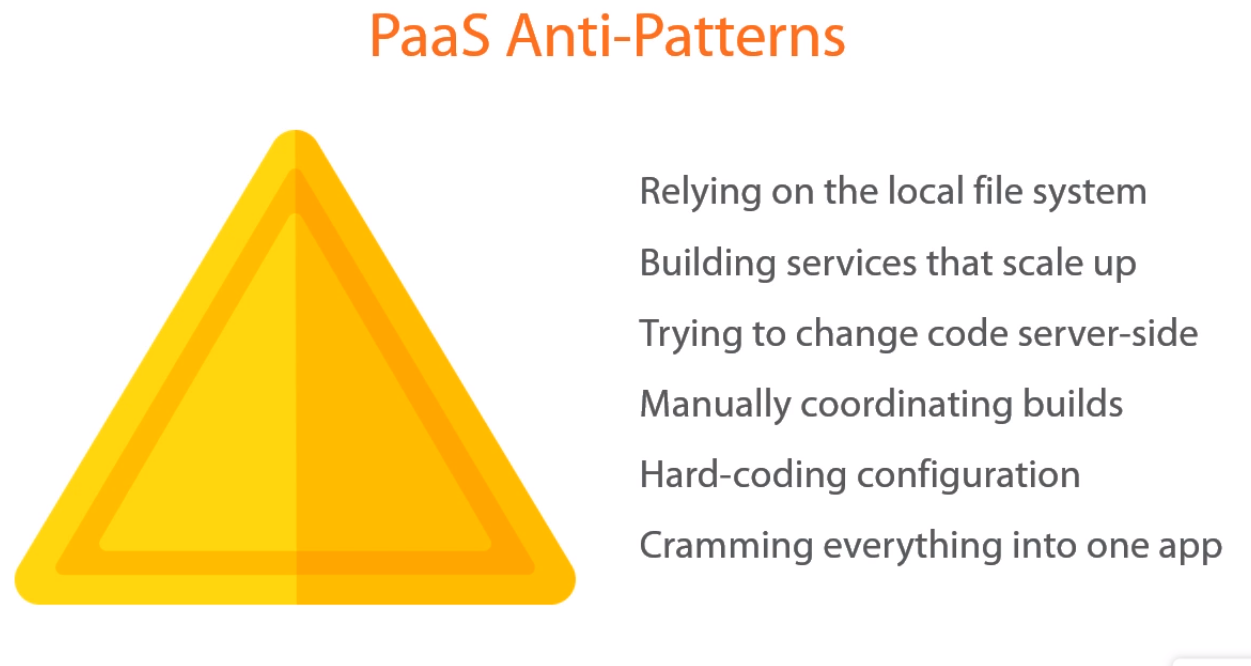
<http://run.pivotal.io> – Sign up for free pivotal account // [stk.rajee@gmail.com/Rajee@3483](mailto:stk.rajee@gmail.com/Rajee@3483)

Navigate to the sample application path in the command prompt and provide the “cf push” command.

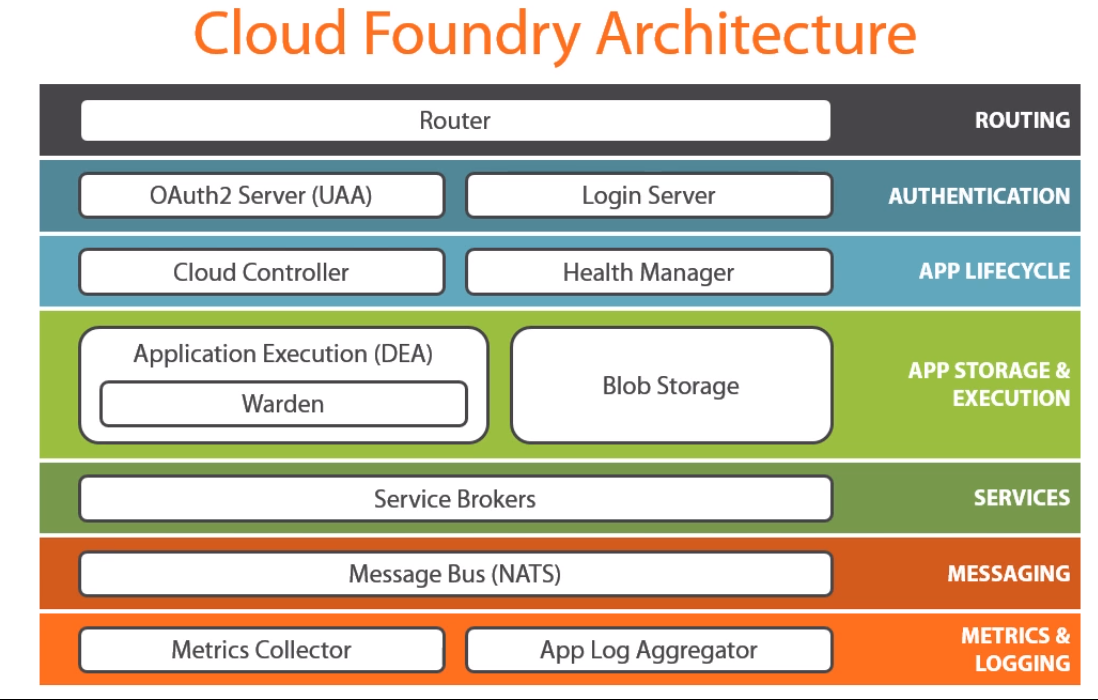
**12 Factor App:**

1. One Codebase tracked in revision control, many deploys[Code base and app should be in 1:1]
2. Explicitly declare and isolate dependencies[Don’t assume that the dependencies will be available in the target. Always provide the required dependencies]
3. Store Config in the environment(Configuration should be stored as environment variables)
4. Treat backing services as attached resources(Local or remote service should be treated in the same manner)
5. Strictly separate build and run stages(Each release should have a unique Id and should be immutable)
6. Execute the apps as one or more stateless processes(Share nothing. Any persistent information should be stored in a backing store)
7. Export services via port binding
8. Scale out via the process model(Work is assigned as a process type, Scaling should be based on OS type process)
9. Maximize robustness with fast start up and graceful shut down
10. Keep Development, staging and production as similar as possible(Gap between the environment should be very minimal or zero)
11. Treat logs as event streams(Application shouldn’t be concerned about the logs. Execution Environment should take care of this like sending the logs to third party tools like Splunk)
12. Run admin/management tasks as one-off processes(Management task should be run from within the environment and not in local so that we get the benefits to see the problems if something goes wrong)

**PAAS Anit Patterns**



**Cloud Foundry Architecture**

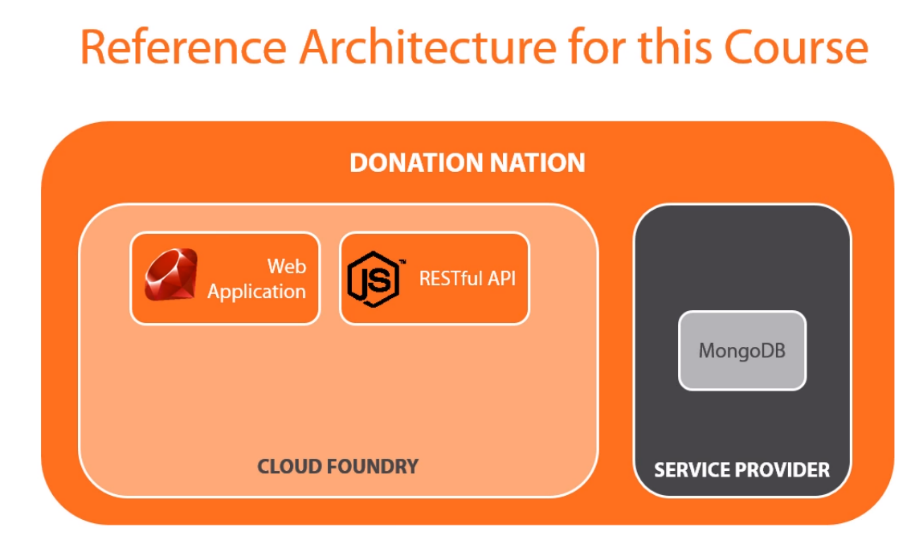
****

All the components within cloud foundry communicates through Message Bus and applications don’t use this. Service Brokers are used for communicating with the third party services like Mysql etc

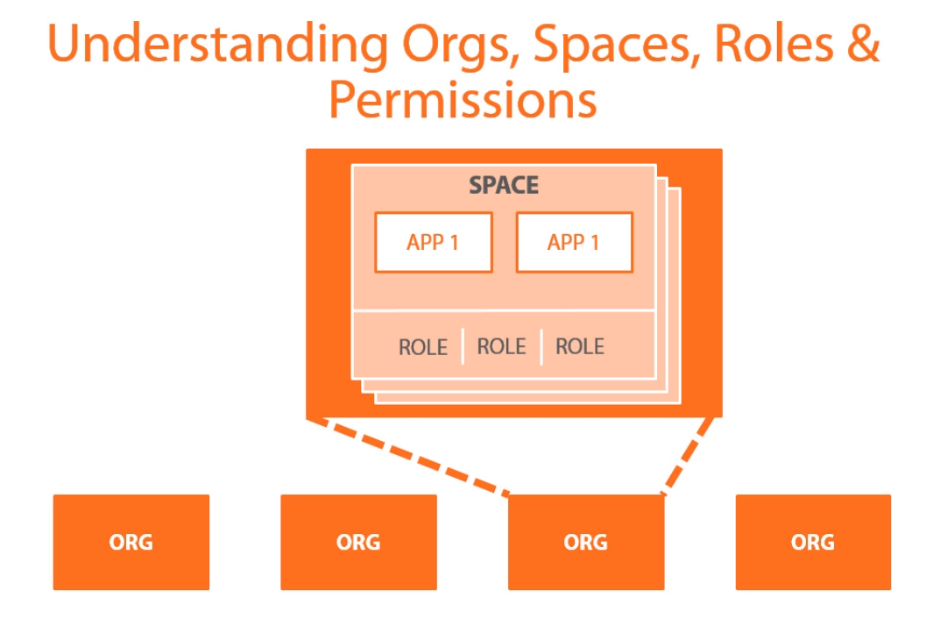
**Cloud Foundry Containers**

Initially they were using LXC as the container. This is linux based, so they moved to Warden which is platform independent. Docker will be integrated in the next version of Cloud Foundry

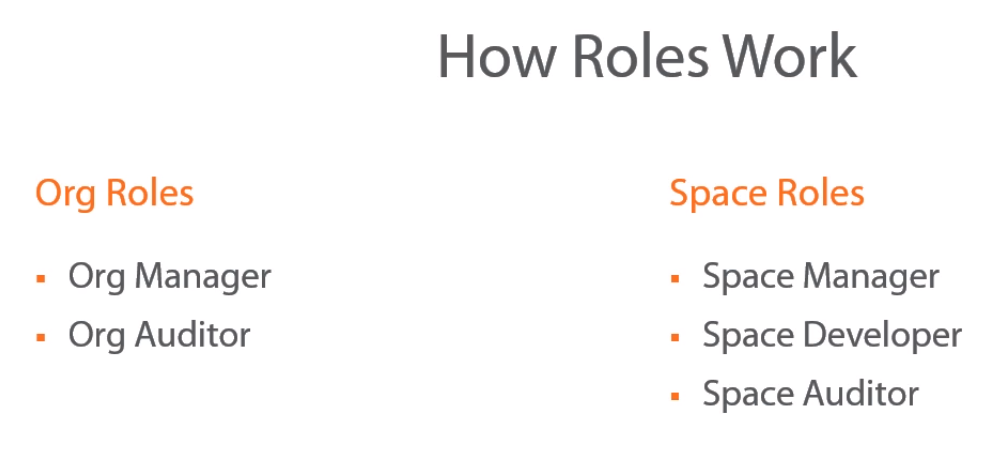
**2. Deploying and Managing Applications Part I**

****

**Understanding Org & Space**

****

**Roles**

****

**Demo**

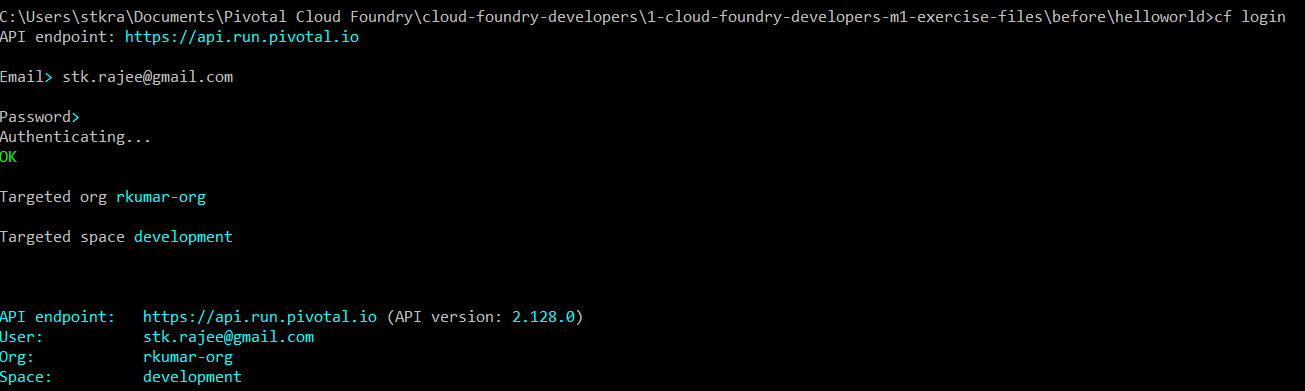
****

Cf – Lists all the CF commands

Cf api – shows the target environment you are connected

Cf api <https://api.run.pivotal.io> – If you have multiple target environment, then you can use this to set the target environment

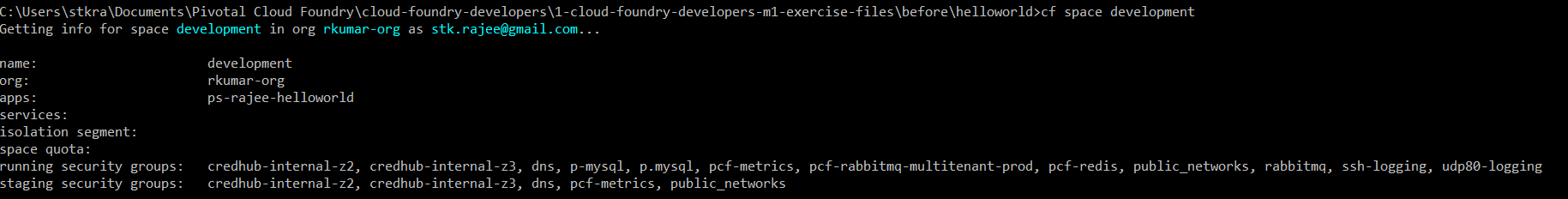
Cf login – This will provide the options to enter the API endpoint, email address and password



Cf spaces – Will list the spaces

Cf create-space <name of space> - This will create a new space

Cf space <space name> - Will list the details for that space



Cf quotas – will list the quotas available for you

Cf target -s <space name> - To switch spaces

<https://login.run.pivotal.io> // To login to pivotal cloud foundry and you can view org, spaces etc..

**Services and Instances**

On demand resources are service instances

Managed Service Instances(All services advertised through Service Brokers), User Provided Service Instances(Services outside cloud foundry)

Cf marketplace – Will list down all the services advertised through service broker

Cf create-service <servicename from market place> <planname> <customized service instance name you want> - To create a service

Cf services – will list all the services

Cf buildpacks – will list all the buildpacks available

**Exploring Manifests**

*Applications:*

Required attribute is name. This is application name

Buildpack – by default, cloud foundry picks up the Heroku build pack model. You can also provide your own.

Stack – environment you wanted to push

Domain – you can specify the domain

Host – subdomain. Domain & subdomain will help build the application route

Command – command to start the application

Instances – This will decide the number of app instances. Each instance will be deployed in a different containers

Memory – memory allocation for this app

Path – will tell where the app is

Timeout – allowable time to start your application

No-route – doesn’t have an internet addressable route. This just runs in background

*Services:*

This will list all the service instances that you want to tie to the application

*Inherit:*

This will be used in child manifest to specify the name of the parent manifest

**What happens during deployment (cf push)**

Cloud controller creates record in its internal db,

CC stores meta data

CC stores app files in its blob store.

App start command is issued.

If its not staged, CC will choose staging DEA based on the information from the build pack and streams the required information if something goes wrong and the developer can troubleshoot it. If things are good, it creates a tar ball also called as droplet and stores it in the blob store. Now the results are cached, so that the information can be used at some later point if needed. Now Staging DEA tells CC that staging is done.

Now CC chooses runtime DEA and the app is ready

**Finish Environment Build Out**

****

Rubyinstaller.org – To install Ruby

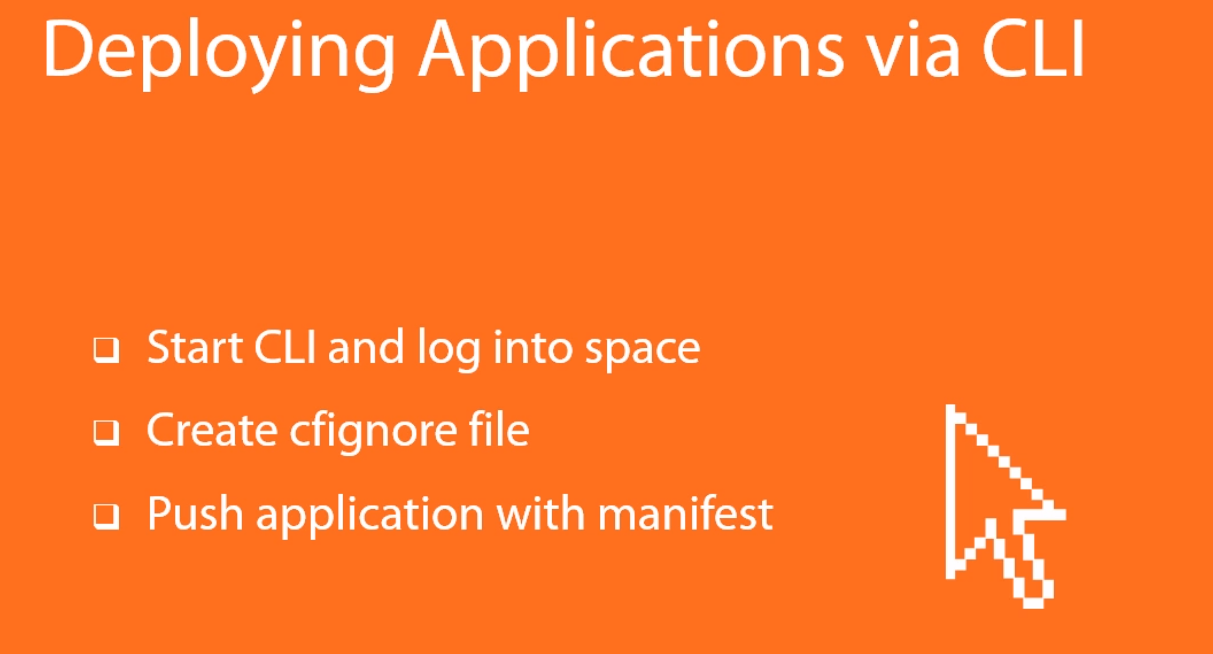
Mongodb.org – To install mongodb

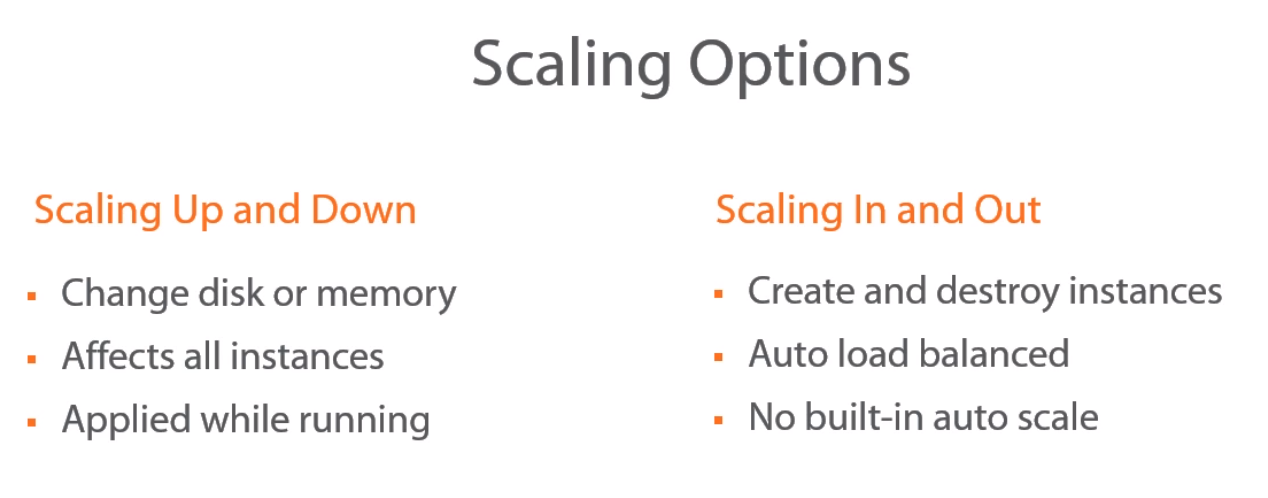
Mongovue.com – To install mongovue

Install Fiddler

Open up a command prompt and start the mongodb. Then go to mongoVue and check if db is started by connecting to the localdb.

**3. Deploying and Managing Applications Part II**





Cf scale <app name> -m 512M // To increase the memory limit

Cf scale <app name> -k 1G // To increase the disk limit

Cf scale <app name> -I 2 // To increase the instance

**Logs:**

Cf logs <app name> /logs // will list down all the log files

Cf events <app name> // List of events that is going on with the application

*Types of logs*

App logs

Stg logs

Router logs

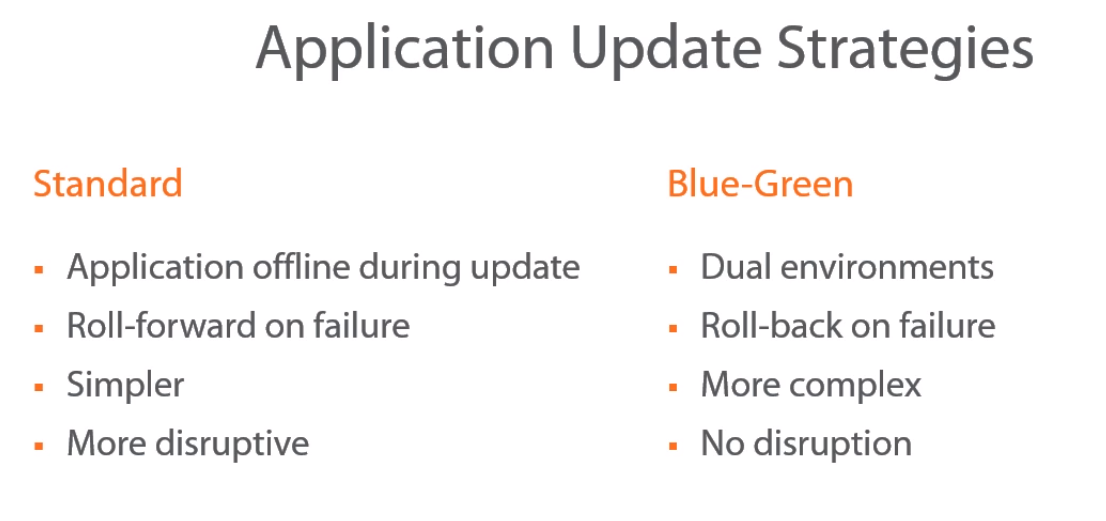
DEA logs

**Environment Variables**

Cf env <appname>

Cf set-env <appname> <variablename> <variablevalue>

Cf unset-env <appname> <variablename>

****

**Application Crash and Recovery**

As soon as one instance goes down, Health Manager sends a message to Cloud Controller to bring up that instance. CC then informs DEA and it brings that instance up