An Overview OF Chef

By- Rashi Sharma

Course Content

✓ Concepts of DevOps

- ❖ Setting Context for DevOps
- ❖ The traditional Dev Vs Ops
- ❖ Explain the spectrum of DevOps
- ❖ Industry Analysis of DevOps

✓ Continuous Integration

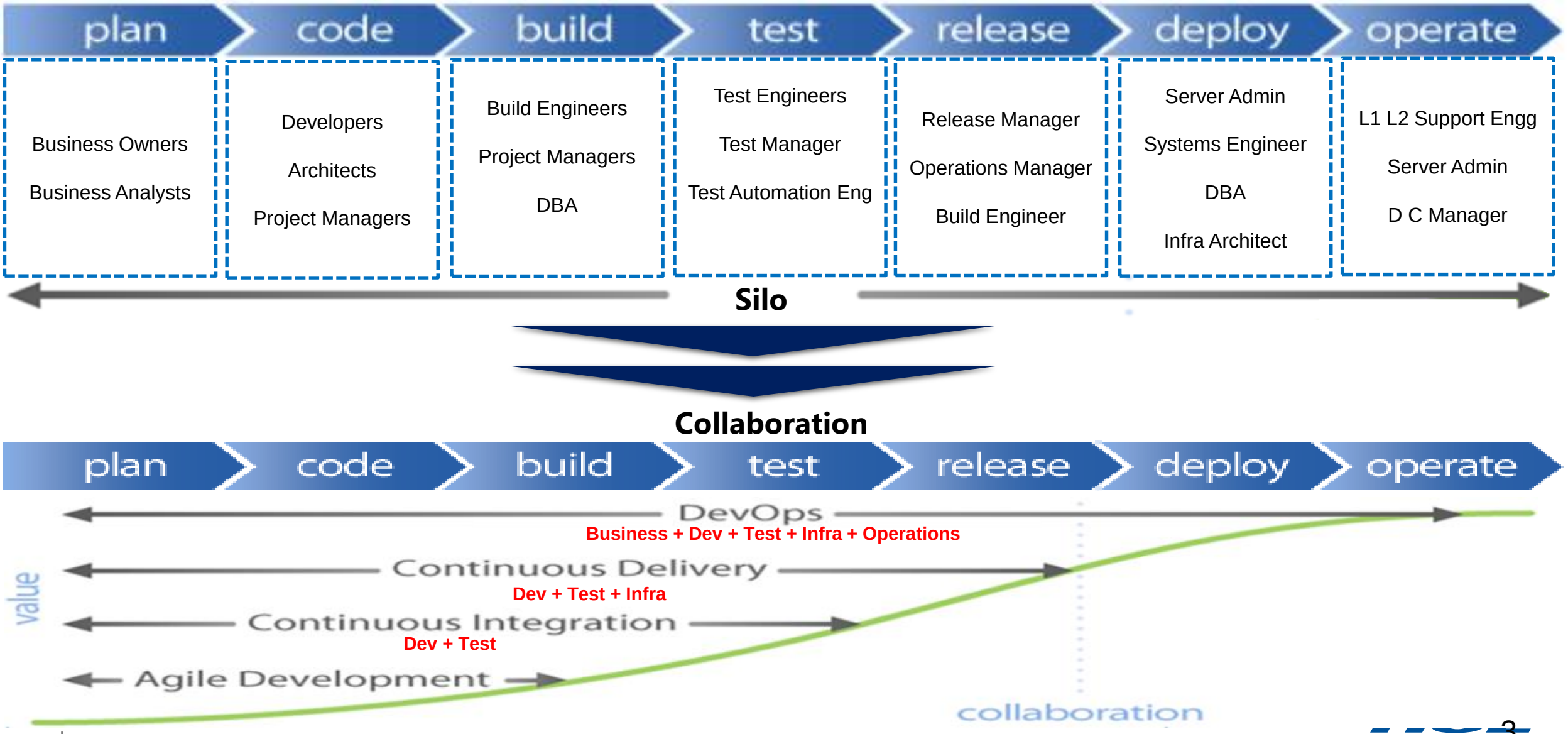
- ❖ Continuous Integration and its benefits to an organization
- ❖ Principles and Practices of Continuous Integration

✓ Continuous Delivery and Continuous Deployment

- ❖ What is Continuous Delivery and Continuous Deployment and their differences
- ❖ Components, tools and workflow of a Continuous Deployment Pipeline

- **Live Demo of an application being build, tested and deployed with single touch CD pipeline**

A paradigm shift in application lifecycle management



A meeting after deployment failure

It was happening again: unable to achieve continuous integration, the DevOps team had fallen back on their less agile strategy, "continuous indignation."

4

Why traditional Dev and Ops conflict

Agile Dev Values:

- Business Driven
- Responsive to Change
- Real Time
- Constantly up to date environment
- CI / CD Environment

Results in:

- Short Sprints (2-3 wk)
- Lots of small changes
- Frequent Deployments

Ops / ITIL Values:

- Procedure Driven
- Stability
- Availability/Uptime
- Controlled/Frozen environment
- Infrequent Updates
- **Results in:**
- Long Lead Time
- Limiting the # of Changes
- Infrequent Deployments

5

What is DevOps?

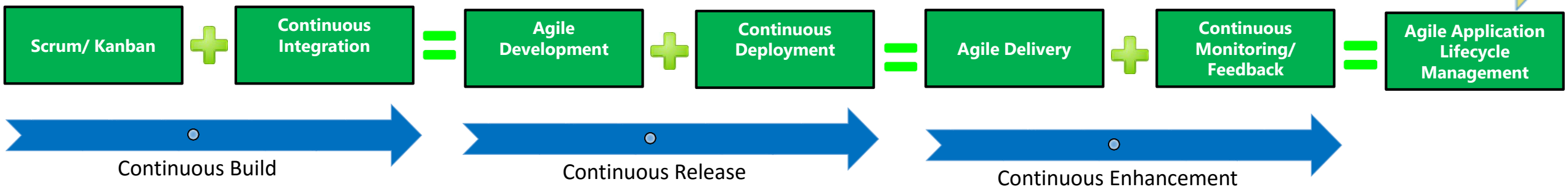
- DevOps is a set of processes and tools that work to achieve higher Profitability, Stability and competitiveness in business.
  - DevOps is a means to improve release frequency, lead time to change and mean time to recover from failure.
    - DevOps is ubiquitous in the any technology, platform and infrastructure.

Business

- Automating the process of build, release and deployment engineering.
- Implementing shift left agile testing and continuous testing into build process.
- Automate the provisioning, configuration and management of infrastructure environment where applications are hosted.

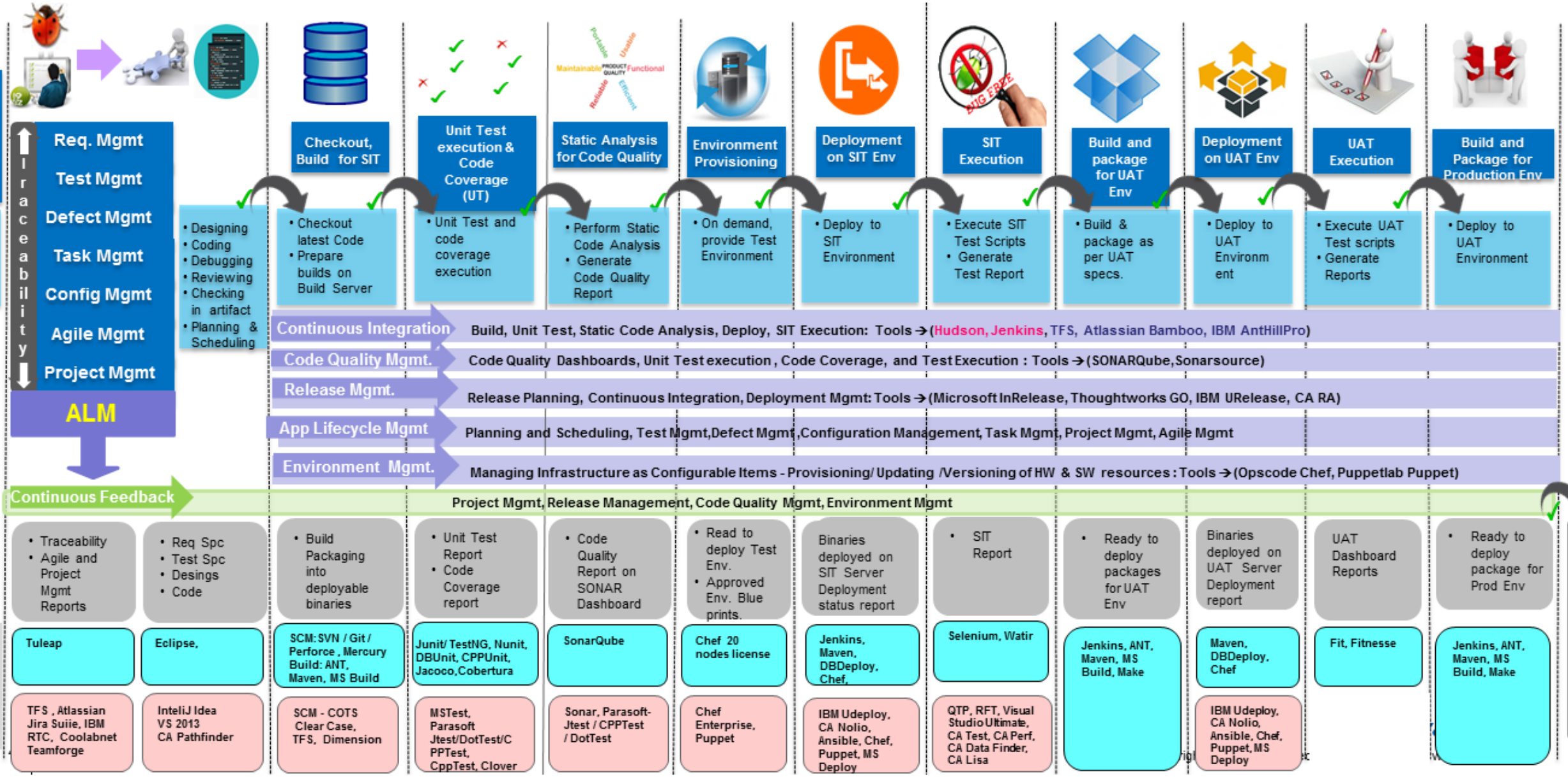
Tech

DevOps processes & tools lead you from Agile Development to Agile Delivery and then to an Agile ALM.



6

Add Agility at each stage of ALM with DevOps



Continuous Integration

Continuous Integration (CI) is a development practice that requires developers to **integrate** code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

Continuous Integration brings multiple benefits to your organization:

- Say goodbye to long and tense integrations.
- Increase visibility which enables greater communication.
- Catch issues fast and nip them in the bud.
- Spend less time debugging and more time adding features
- Proceed in the confidence you're building on a solid foundation
- Stop waiting to find out if your code's going to work.
- Reduce integration problems allowing you to deliver software more rapidly.

8

Continuous Integration: Principles and Practices

The Practices

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Make it easy for anyone to get the latest executable
- Everyone can see what's happening
- Automate deployment

Team Responsibilities

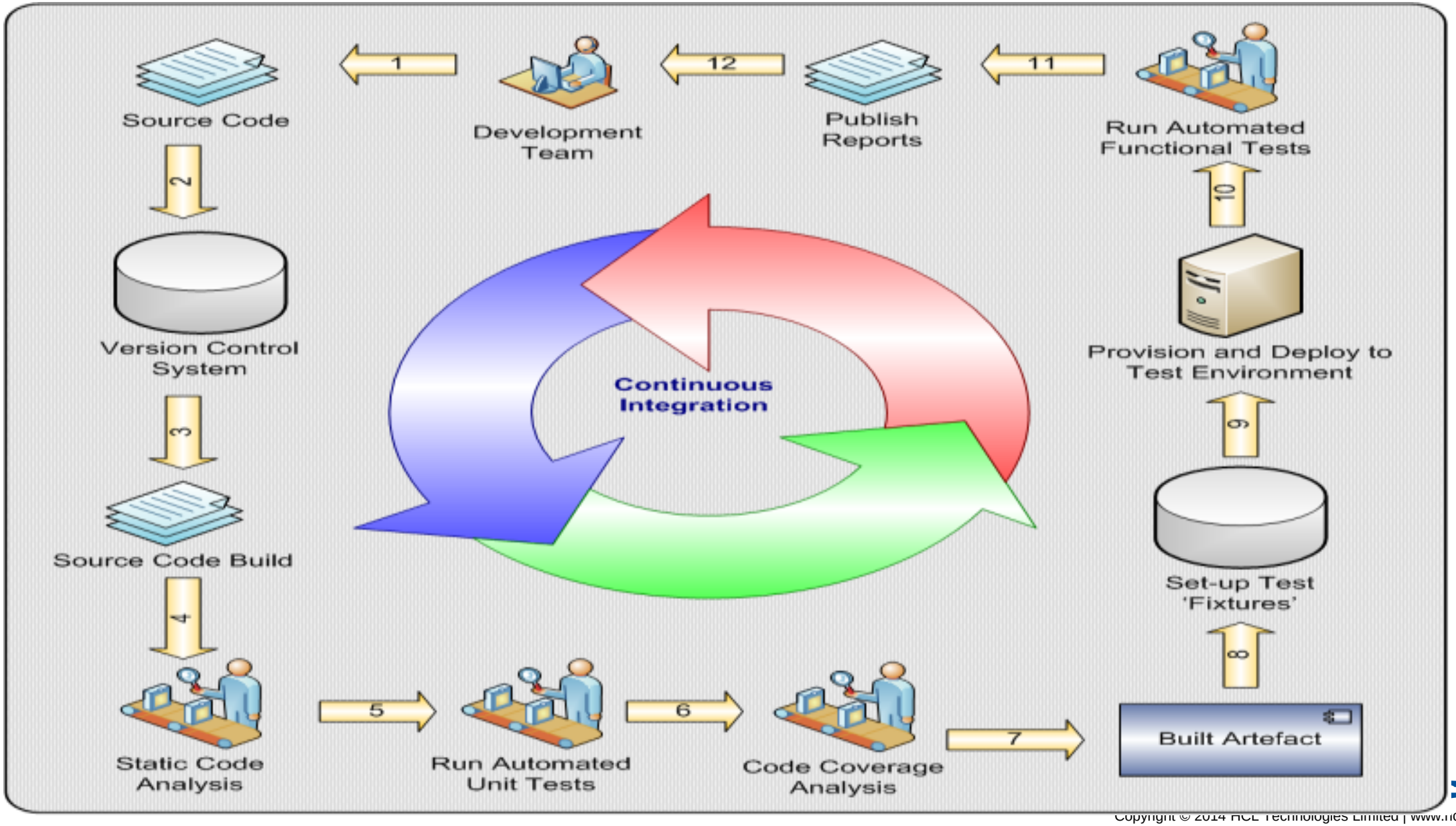
- Check in frequently
- Don't check in broken code
- Don't check in untested code
- Don't check in when the build is broken
- Don't go home after checking in until the system builds

How to do it

- Developers check out code into their private workspaces.
- When done, commit the changes to the repository.
- The CI server monitors the repository and checks out changes when they occur.
- The CI server builds the system and runs unit and integration tests.
- The CI server releases deployable artefacts for testing.
- The CI server assigns a build label to the version of the code it just built.
- The CI server informs the team of the successful build.
- If the build or tests fail, the CI server alerts the team.
- The team fix the issue at the earliest opportunity.
- Continue to continually integrate and test throughout the project.

9

Continuous Integration Workflow



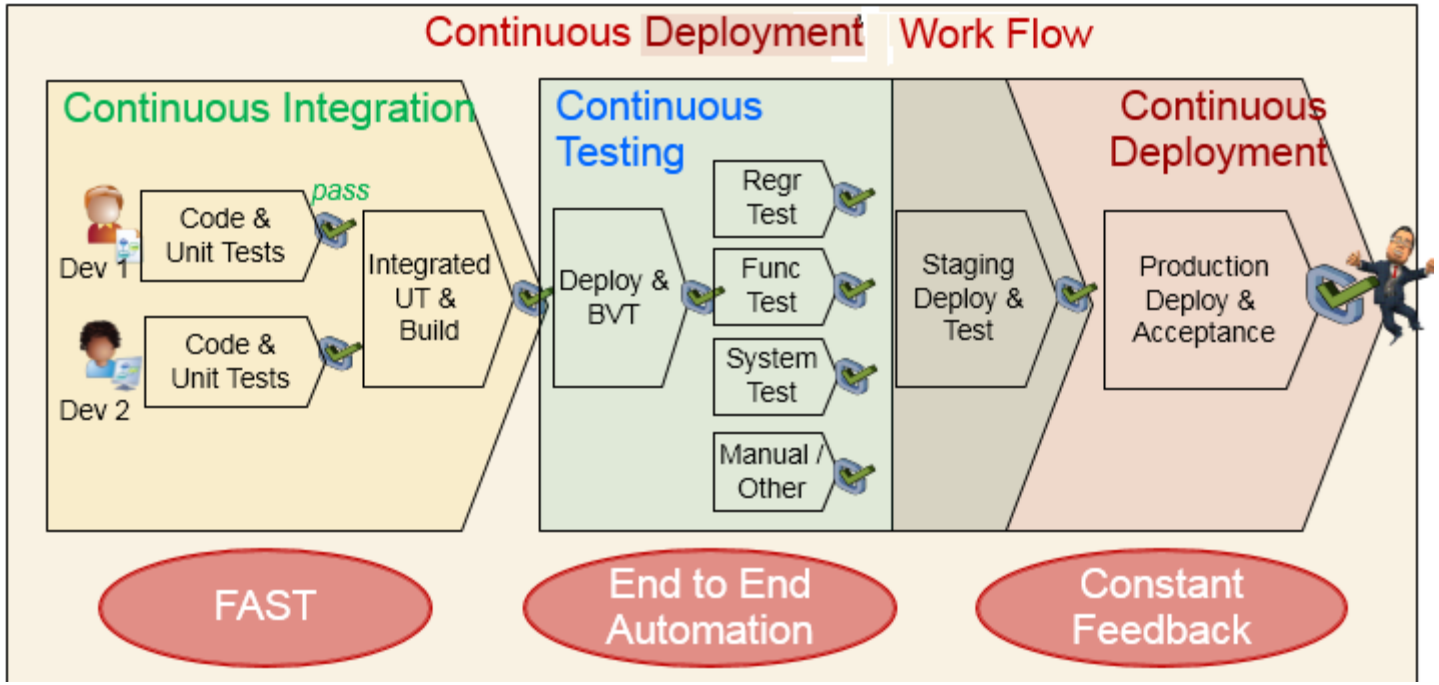
Continuous Delivery

Continuous delivery (CD) is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way. Benefit is ***faster time to market***.

11

Continuous Deployment

Continuous Deployment can be thought of as an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by live users, in production.



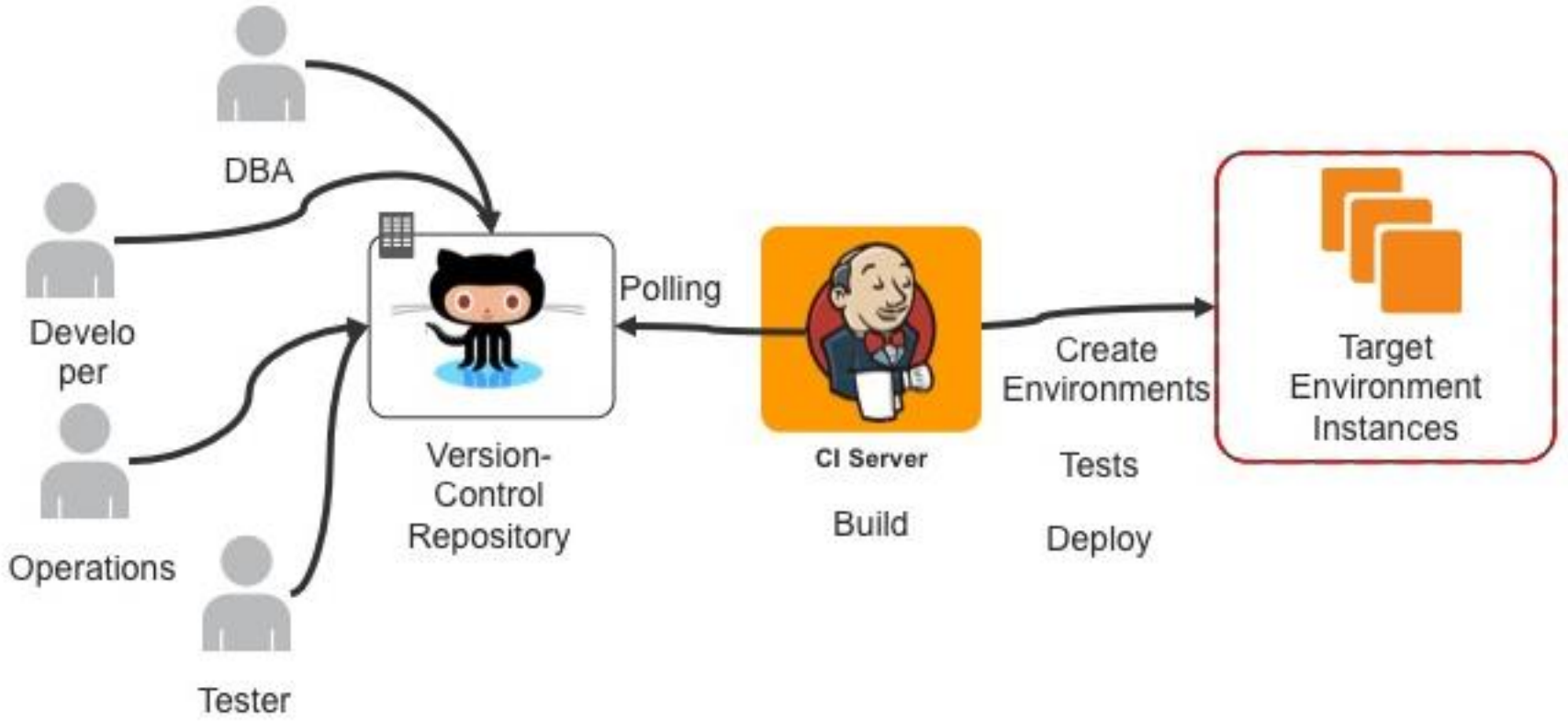
12

Jenkins

Jenkins is an open source continuous integration **tool** written in Java. **Jenkins** provides continuous integration services for software development. It is a server-based system running in a servlet container such as Apache Tomcat.

Continuous Integration and Continuous Delivery

- As an extensible [illegible] any project.



Jenkins Jobs

Now lets start creating new job:

- click *New Item* on the left
- type the name of the job
- check *Freestyle project*
- click *OK*

You should be redirected to configuration page of your job. Lets create step definitions for our job:

- under source code management select Git
- under Repository URL paste following: https://github.com/<group>/<Repository.git>
- in Build section click on the Add build step button and select: execute shell
- in the command text box paste the command to be executed e.g “javac HelloWorld.java”
- click Save

Jenkins Plugins

- Once Jenkins is installed, minimally configured, and reasonably secured, it's time to make it fit your needs. As found when it is first installed, Jenkins has relatively few abilities.
- Plugins are add-ons that allow Jenkins to interact with a variety of outside software or otherwise extend its innate abilities. As with many areas of the Jenkins setup, the exact plugins you install will be significantly dependent on your projects.
- The plugins eliminate the need to create fancy, half tested logic to accomplish tasks, solving common problems with minimal pain and also promote reusability across projects.
- Jenkins installs some of plugins by default
  - CVS, Subversion, Ant, Maven, Javadoc, Junit etc. for development
- Plugins are grouped based on the functionality: Source code management, Build triggers, Build tools, UI, Authenticators etc.
- Plugins related to the project are installed and configured.

Infra Configuration Management

- A configuration management system gives you the ability to deploy, update, and repair your entire application infrastructure using pre-defined, automated procedures.
- You can get the information that typically includes the versions and updates that have been applied to installed software packages and the locations and network addresses of hardware devices.
- When a system needs a hardware or software upgrade, a computer engineer can access the configuration management program and database to see what is currently installed. He can then make a more informed decision about the upgrade needed.
- An advantage of a configuration management application is that the entire collection of systems can be reviewed to make sure any changes made to one system do not adversely affect any of the other systems.
- Ideally, you want to automatically provision your entire environment from bare-metal all the way up to running business services completely from a pre-defined specification, including the network configuration.

Introduction to Chef

Think about some of the tasks you might do repeatedly:

- ❑ Install an operating system on a new computer
- ❑ Upgrade the operating system on a new computer
- ❑ Install software libraries
- ❑ Install Apache
- ❑ Modify an existing Apache configuration

Using Chef, you can write simple scripts (called recipes in Chef parlance), and then Chef will handle the automation for you.

Configuration management tool written in Ruby and Erlang

Chef automates the applications configuration, deployment and management, even if you're operating in the cloud, on-premises or on hybrid.

Chef converts infrastructure into code

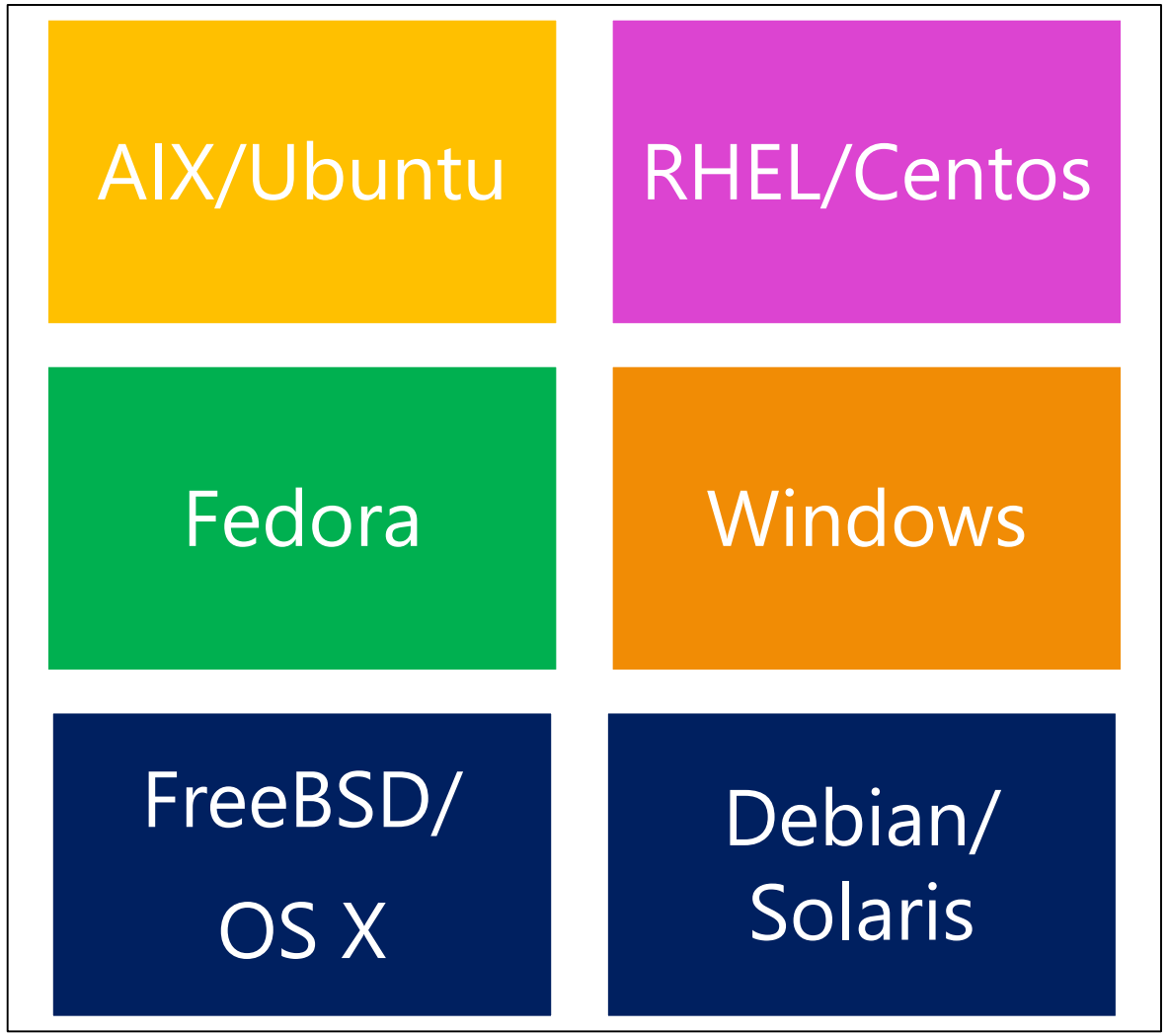
It make your infrastructure easily versionable, testable and repeatable as the application code.

It can automate the server configuration throughout your network regardless of its size

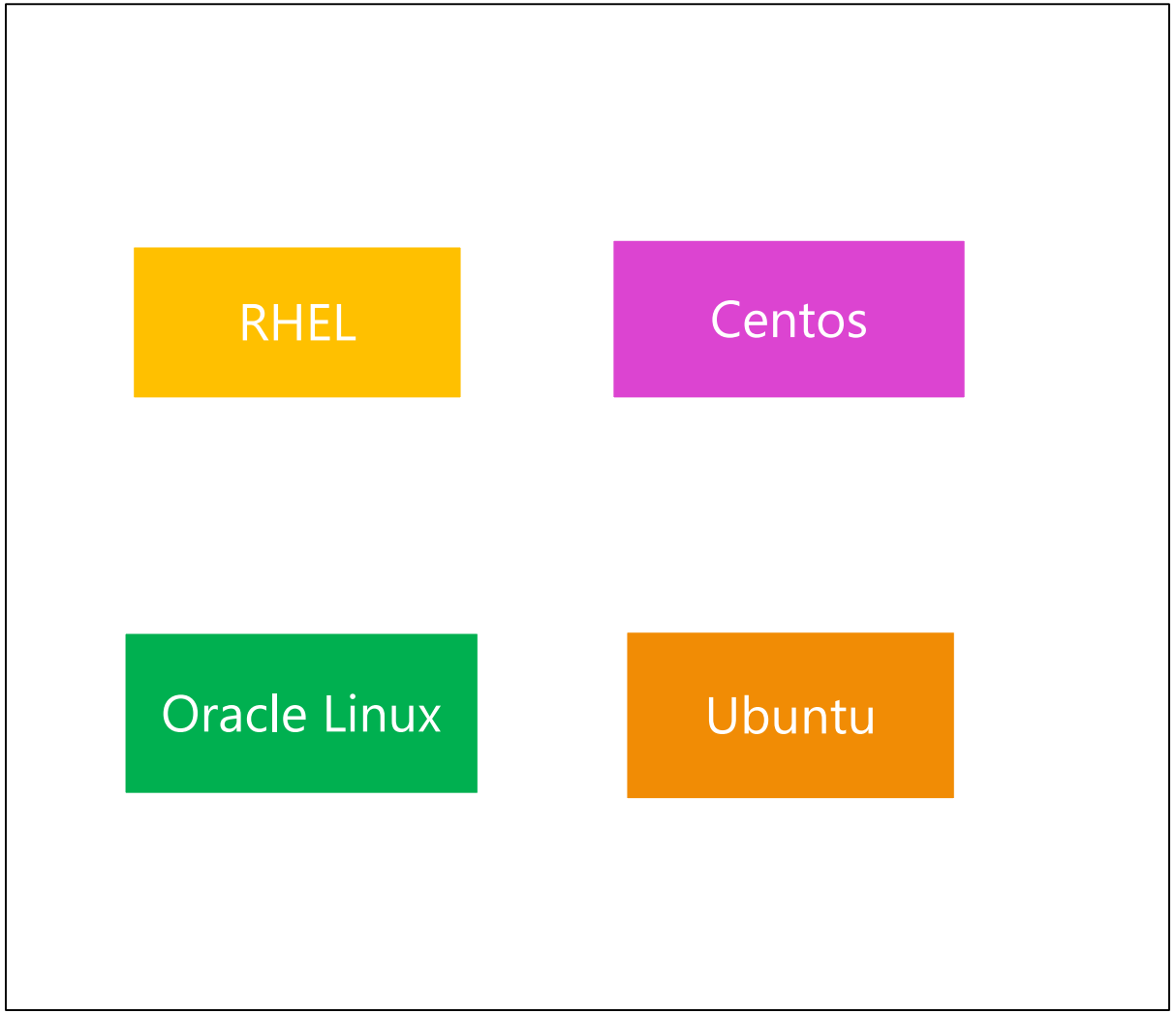
17

Supported Platforms

- Major platform support for Chef-client includes



Chef Server is supported on.



Chef Integration with Cloud Based Platforms

Chef can integrate with cloud-based platforms such as:

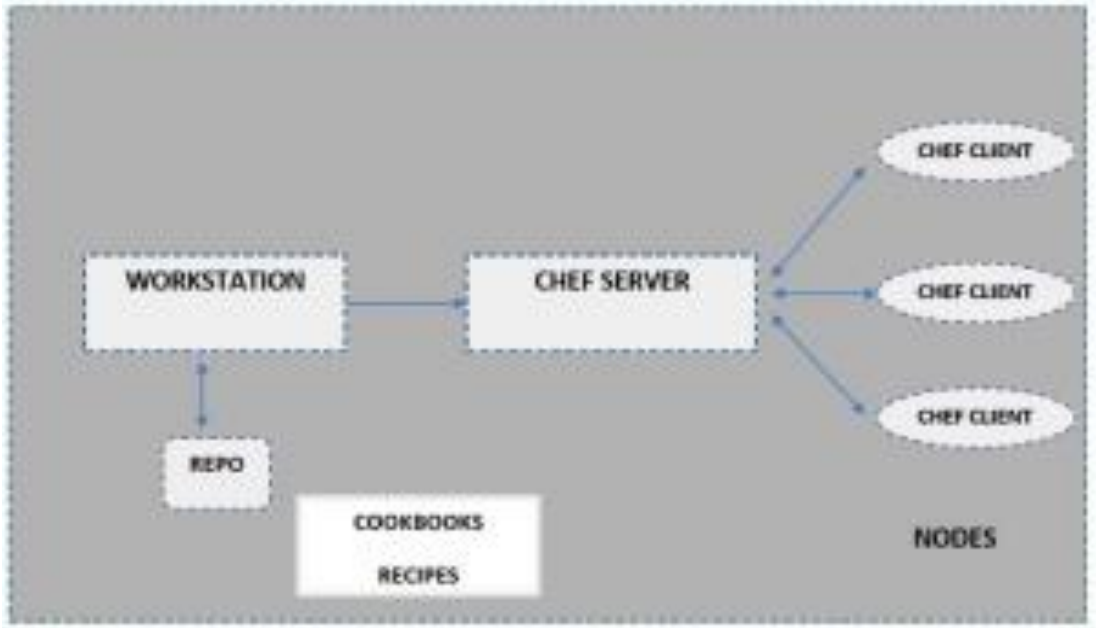
- Amazon EC2
- OpenStack
- Microsoft Azure
- Google Cloud Platform
- Rackspace

Chef Components

Chef Server: The Chef Server stores and holds all the cookbooks, recipes and metadata that illustrates each and every registered node which is being managed by the chef-client and the policies that are applied to nodes.

Workstation: Workstation is the place where user will spend most of their time with Chef and will do most of their work that includes:

- Development of the cookbooks and the recipes
- Making the use of knife to upload items from the chef-repo to the Chef server
- Configuring organizational policies i.e. defining roles as well as environments.
- Communicating with the nodes whenever needed, such as carrying out a bootstrap operation.



Chef Nodes: A node can be a physical, virtual or cloud machine that contain the chef-client.

- Nodes are the computers that we manage using chef and it act as server in our infrastructure.

Getting Started With Chef

First Install and configure the Chef server

Then install and configure a workstation

Run the bootstrap command from the workstation to install the chef-client on each node.

21

System Requirements

Hosted Chef Server

The hosted Chef server has the following requirements:

- **Browser** — Firefox, Google Chrome, Safari, or Internet Explorer (versions 9 or better),Create an acoount on https://api.chef.io.
- Every node having the chef-client and every workstation that will upload data to the Chef server must be able to communicate with the hosted Chef server.

Chef-node- To practice locally we can download a Linux/windows ISO and configure virtual box to run it. This will act as chef node

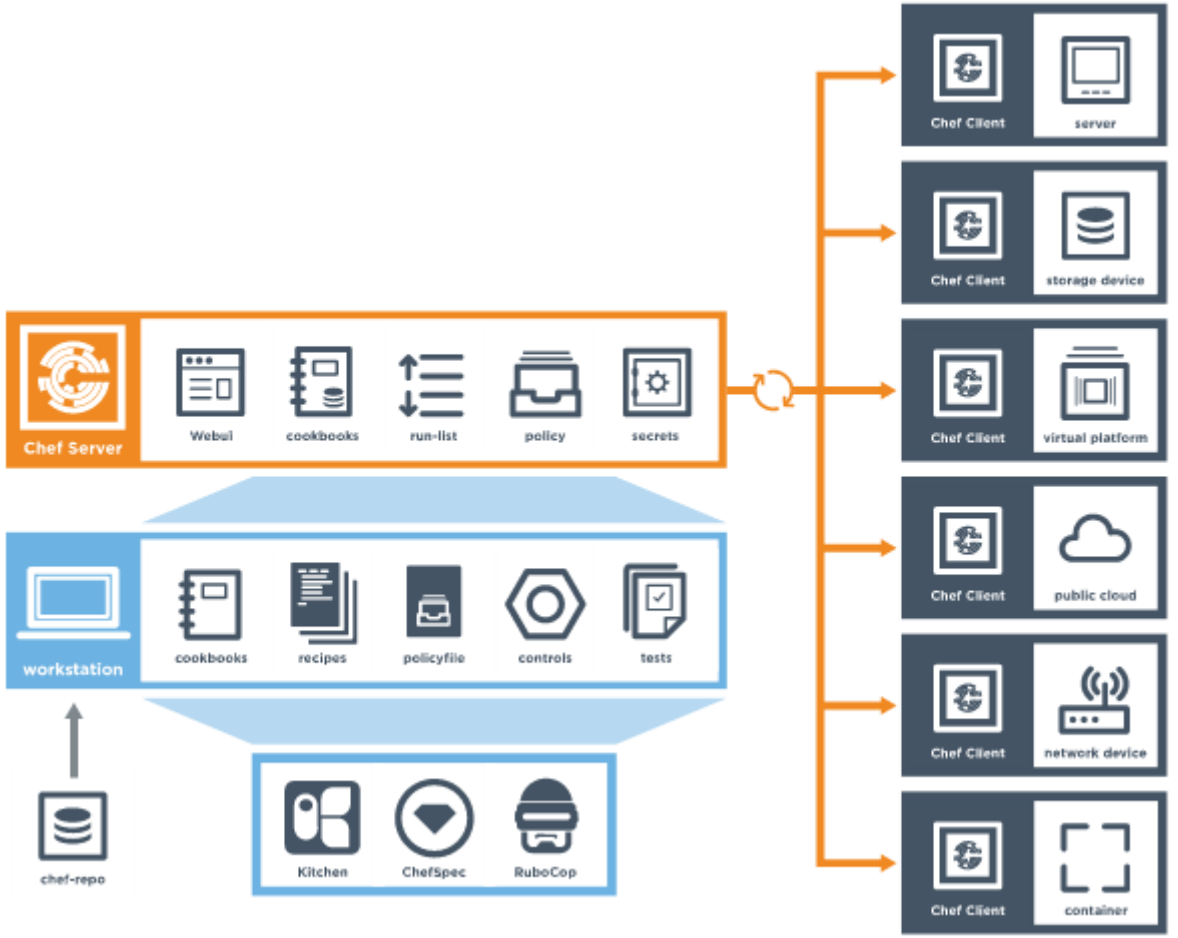
- The recommended amount of RAM available to the chef-client during a chef-client run is 512MB
- The chef-client binaries requires a minimum of 200MB of disk space
- The chef-client caches downloaded cookbooks, packages required by those cookbooks, and other large files during the chef-client run. This directory (/var/chef/cache) requires atleast 5 GB of space to save all of this data at the starting point.
- Each node and workstation must have access to the Chef server via HTTPS.
- Ruby 2.2.2 (or higher).
- The hosted Chef server is compatible with chef-client version 0.10.0 and greater.

Workstations(You can configure it on your local desktop/laptop**)**

- Knife
- Knife.rb: A knife.rb file is used to specify the chef-repo-specific configuration details for knife.
- Chef-repo

How Chef Works

- The user writes "recipes" that describe how Chef manages server applications and utilities (such as Apache HTTP Server, MySQL, or Hadoop) and how they are to be configured like packages that should be installed, services that should be running, or files that should be written.
- The Chef server holds up all the recipes as well as other configuration data.
- Now the Chef client is installed on all the targeted nodes in your network.
- The Chef client at regular intervals polls the Chef server for the most recent recipes and examines whether the nodes is in compliance with the policy defined by these recipes.
- If a situation arises where a node is outdated then the Chef client runs on those node so as to bring it up to date.



Chef Workstation Setup

We will now setup a workstation so you can edit you Chef recipes in your favourite text editor.

Install the Chef DK from https://downloads.chef.io/chef-dk

Install git from https://help.github.com/articles/set-up-git/

Set up the chef-repo

Cookbook & Its Components

- A cookbook is the fundamental unit of configuration that Chef uses to bring a node into a specific state.
- Chef uses cookbooks to perform work and make sure things are as they should be on the node.

Recipe

- A recipe is the most fundamental configuration element for a node

Template

- A cookbook template is an Embedded Ruby (ERB) template that is used to dynamically generate static text files.

Metadata

- Every cookbook requires a small amount of metadata. A file named metadata.rb is located at the top of every cookbook directory structure.

Files

- Use the **cookbook_file** resource to transfer files from a sub-directory of COOKBOOK_NAME/files/ to a specified path located on a host that is running the chef-client.

Attributes

- An attribute can be defined in a cookbook COOKBOOK_NAME/attributes/default.rb file and then used to override the default settings on a node.

Chef Client Setup

- A chef-client is an agent that runs locally on every node that is under management by Chef. When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state, including:
- Registering and authenticating the node with the Chef server
- Building the node object
- Synchronizing cookbooks
- Compiling the resource collection by loading each of the required cookbooks, including recipes, attributes, and all other dependencies.
- Taking the appropriate and required actions to configure the node.
- Looking for exceptions and notifications, handling each as required.

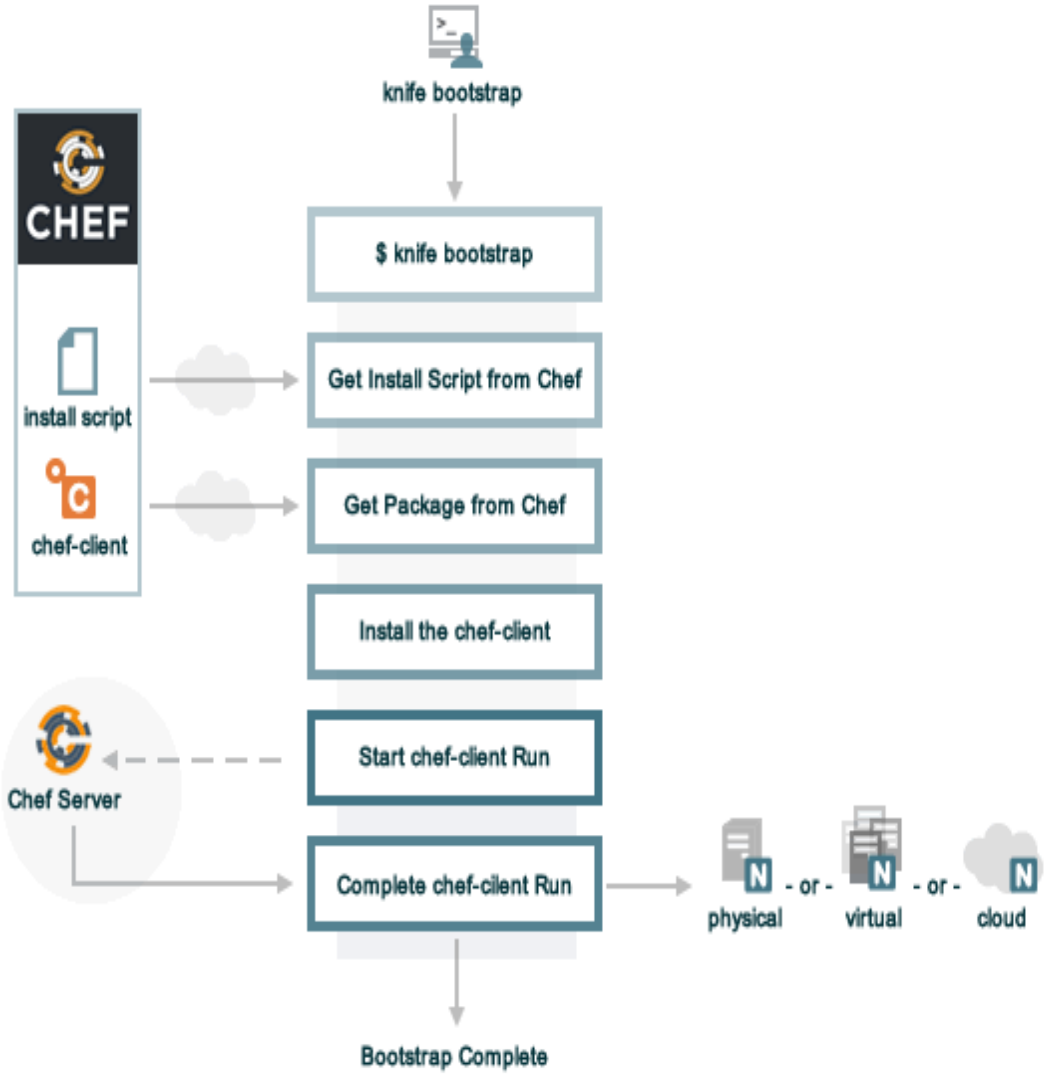
Bootstrapping a node with Knife

A bootstrap is a process that installs the chef-client on a target system so that it can run as a chef-client and communicate with a Chef server.

knife bootstrap

- The knife bootstrap command is a common way to install the chef-client on a node.
- The omnibus installer will detect the version of the operating system, and then install the appropriate version of the chef-client using a single command to install the chef-client and all of its dependencies, including an embedded version of Ruby, RubyGems, OpenSSL, key-value stores, parsers, libraries, and command line utilities.
- In a command window, enter the following:

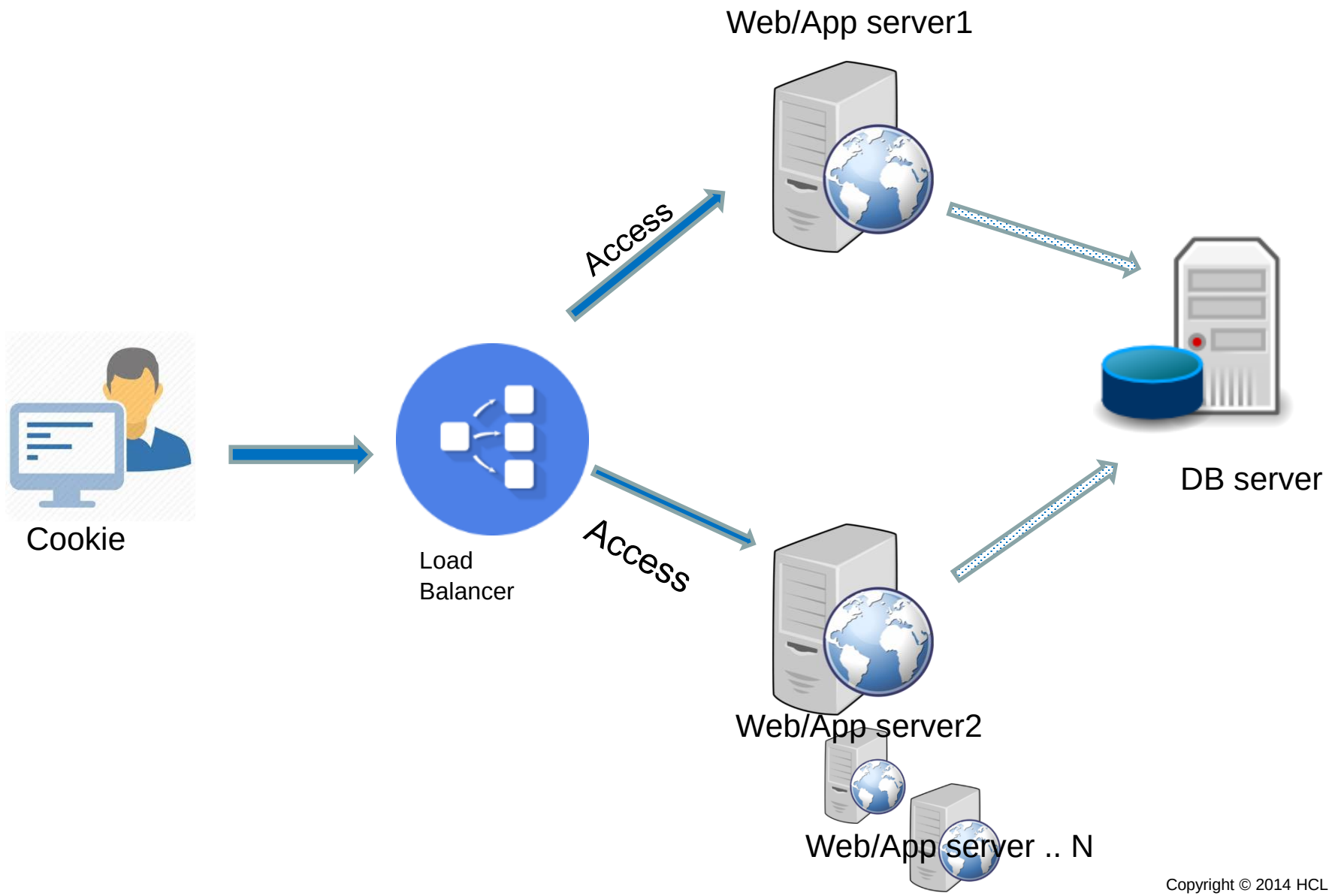
Knife bootstrap <FQDN> -x username –P password –sudo



Live Demonstration of Continuous Integration and Continuous Deployment

28

Application Architecture



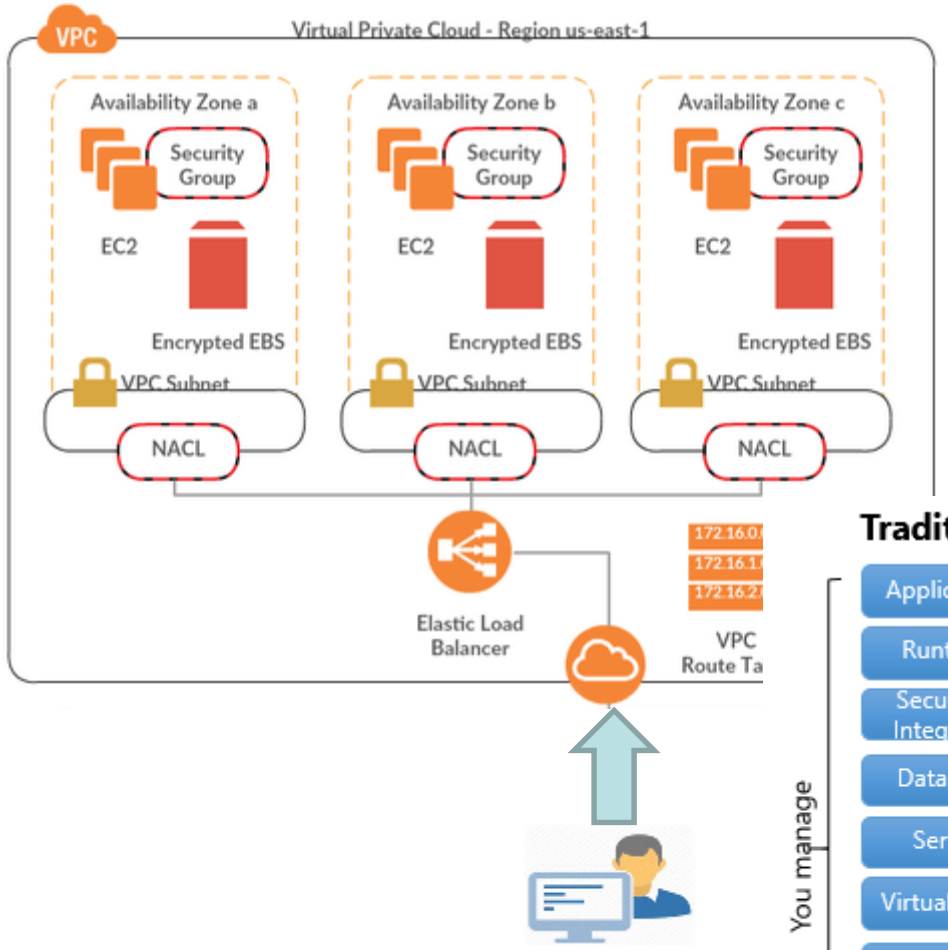
29

Example: Web Application

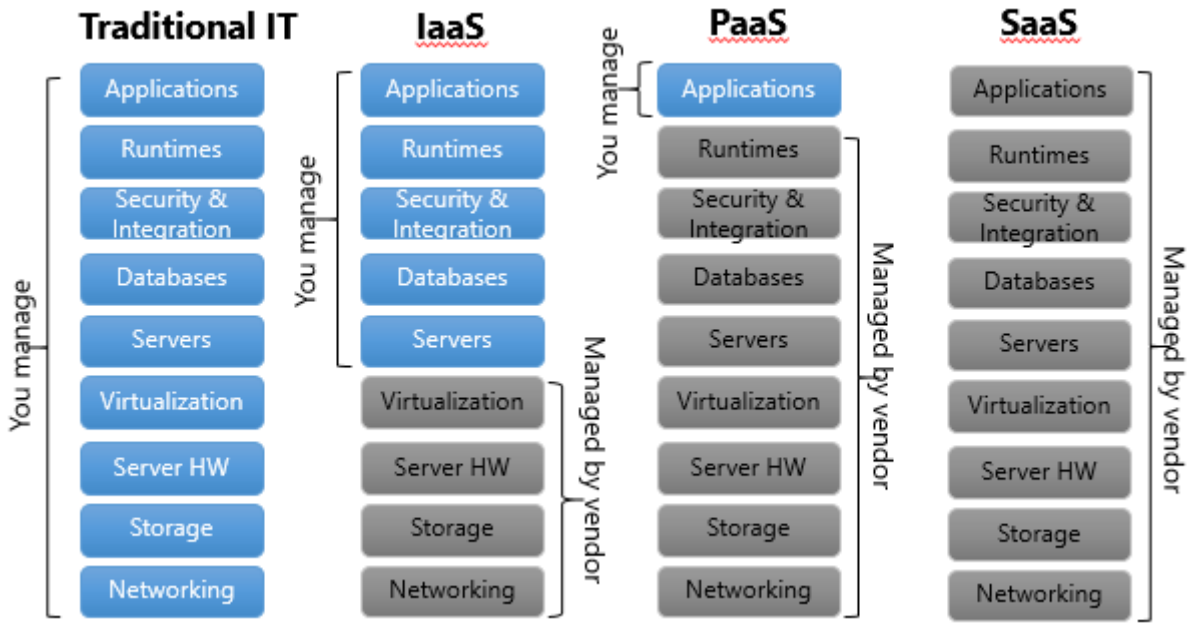
- DB - mysql
  - Uses database “UserDB”,
  - Table “Users” and User “Pankaj”
- Web application - Tomcat as Server
  - HTML, Servlet and Jsp
  - Registers the user details in the DB
  - Retrieves the details for login, credentials checks
  - Serve the pages once authenticated
- Logging
  - Log every transaction and custom configuration file

30

AWS Cloud Component & Application Architecture



Accessing a HA Website On AWS Cloud



What is CloudOps?

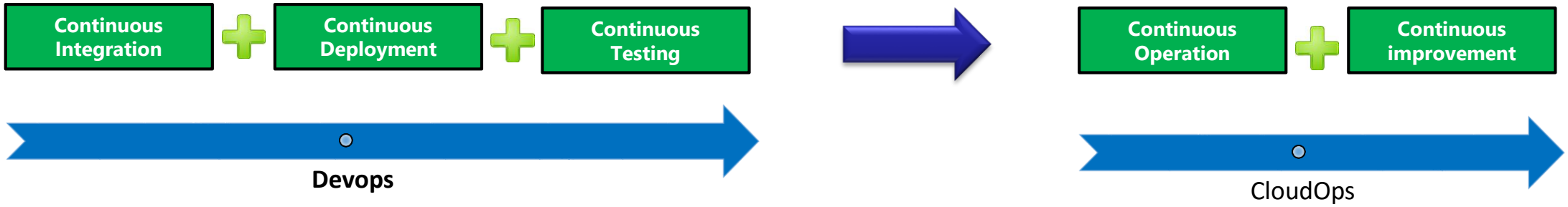
- Cloudops is hugely valuable to the business, considering that the objective of CloudOps is an operation that never, ever stops.
- CloudOps is based on the idea of continuous operations & continuous improvement.
  - Objective of CloudOps is zero downtime.

Business

- CloudOps designs the next generation of cloud based network architectures.
- CloudOps works to create transient, disposable, permanent and semi-permanent IT and network resources, using networking technologies built for the cloud.
- Automate the operations activity, deployment of code on infrastructure environment where applications needs to be hosted.

Tech

CloudsOps process helps achieve continuous operations in public & private cloud as the end state of continuous development, testing,



Thank You

DevOps COE

To know more about exciting new things in DevOps at HCL

Contact- Nitin Shah (shah.ni@hcl.com)

HCL