

Configuration Management And Automation



Atul Kumar
Author & Cloud Expert



Atul Kumar

LinkedIn QR code

Scan

My code



Atul Kumar

Founder at K21Academy: Learn Cloud
From Experts



- Author & Cloud Architect
- 21+ Years working in IT & Certified Cloud Architect
- Helped **6000+ individuals** to learn Cloud & Cloud Native

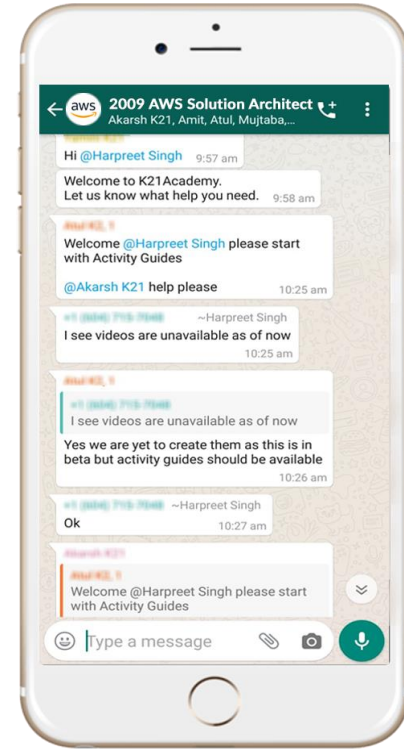


ORACLE
ACE



WhatsApp & Ticketing System

support@k21academy.com





Module Agenda

Agenda: Module

- Need of CloudFormation
- CloudFormation and its components
- Templates in CloudFormation
- Stack in CloudFormation
- Installation of LAMP server in EC2 via CloudFormation
- AWS OpsWorks and its Lifecycle events
- AWS Ops Works services
- OpsWorks for Chef Automate
- Cookbooks and Recipes

Agenda: Module

- Ops Works for Chef Automate
- Cookbooks and Recipes
- AWS Ops Works Stack
- Components of AWS Ops Works Stack
- AWS OpsWorks for Puppet Enterprise
- Elastic Beanstalk
- Components of Elastic Beanstalk
- Beanstalk v/s OpsWorks v/s Cloud Formation



CloudFormation

CloudFormation

AWS CloudFormation automates and simplifies the task of repeatedly and predictably creating groups of related resources that power your application.

01

It provides an easy way to create and manage a collection of AWS resources

02

Here you can create template of your application, so as to have its copies for backup

03

It enables us to manage our AWS resources or complete infrastructure via a text file

04

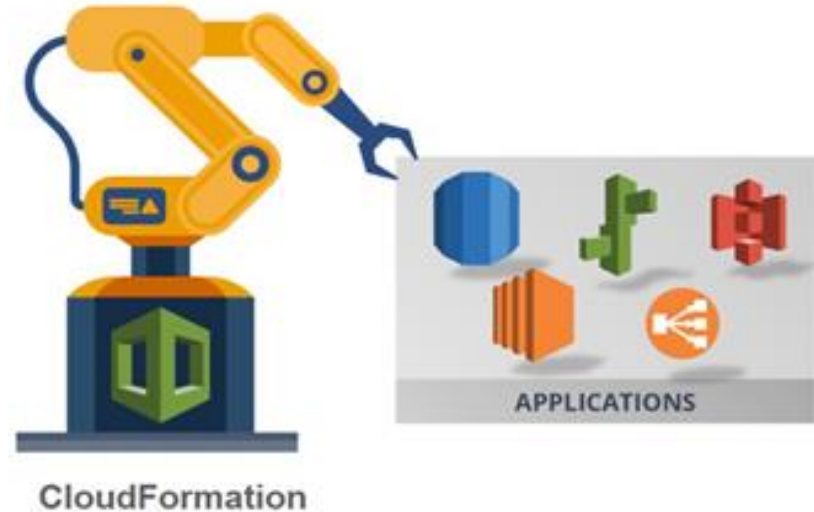
You can deploy and update stack using CLI, Console or API



AWS CloudFormation

Why Do We Need AWS CloudFormation

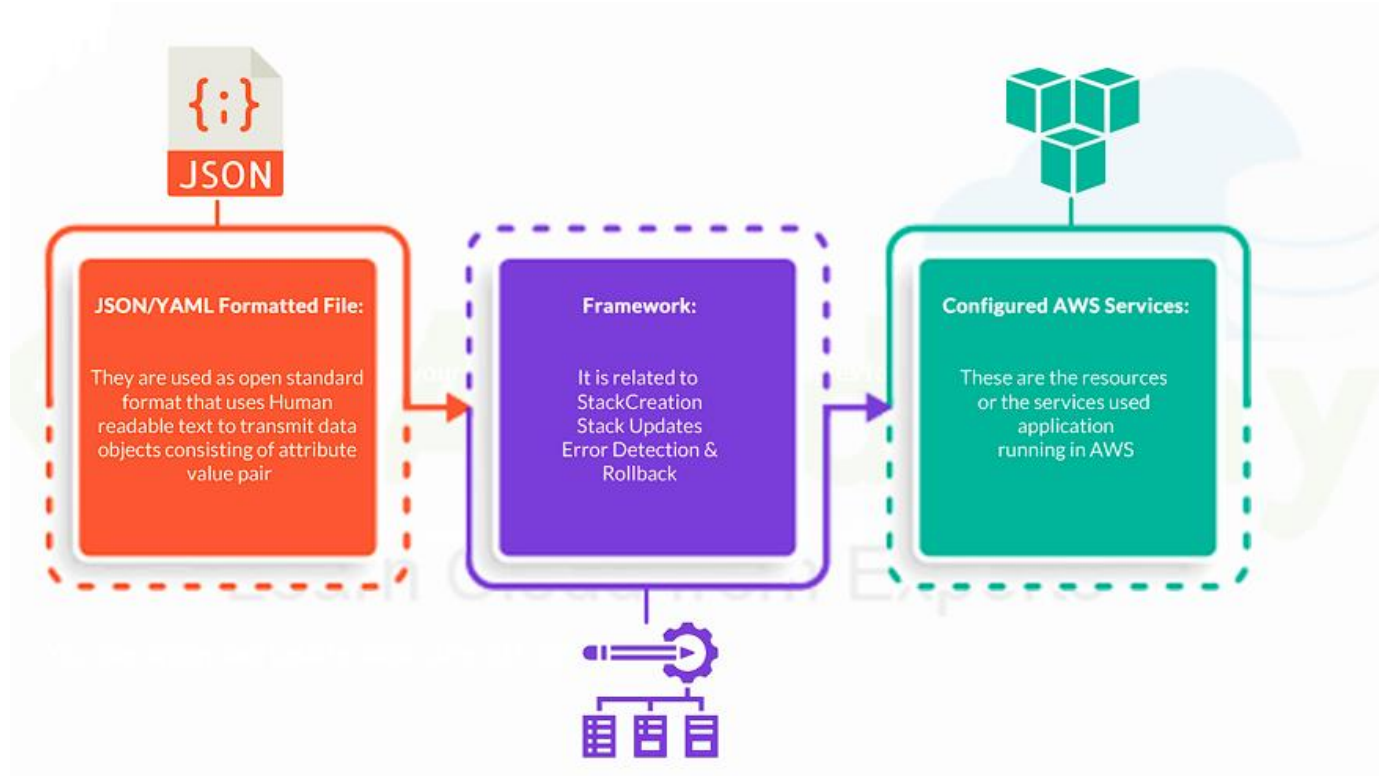
- An application on the AWS may require a certain set of resources and managing these resources is not an easy task even it consumes more time
- This management part can be skipped if we make use of AWS CloudFormation
- Hence, using CloudFormation we can manage, create and provide all these resources at the same place





Components Of CloudFormation

CloudFormation - Components



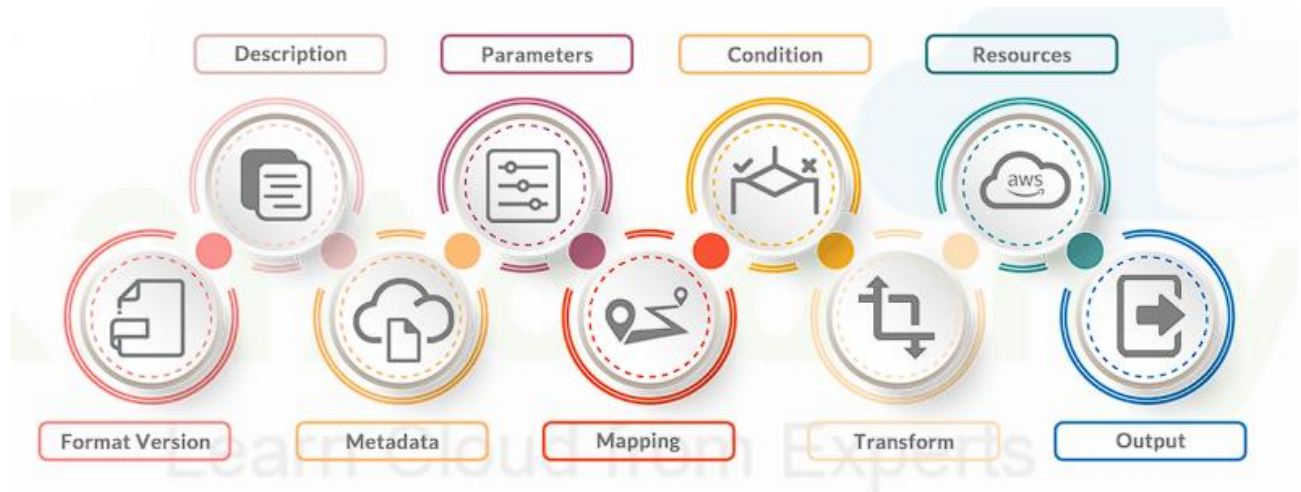


Templates and Stack In CloudFormation

Templates In CloudFormation

Templates in AWS CloudFormation are the formatted text files represented in JSON or YAML which describes your AWS Infrastructure Using AWS CloudFormation designer or any text editor tool you can create, modify and view these templates

Major element of
AWS CloudFormation
Templates:



Templates Structure

```
{
  "Description" : "A text description for the template usage",
  "Parameters": {
    // A set of inputs used to customize the template per deployment
  },
  "Resources" : {
    // The set of AWS resources and relationships between them
  },
  "Outputs" : {
    // A set of values to be made visible to the stack creator
  },
  "AWSTemplateFormatVersion" : "2010-09-09"
}
```


Sample Template – Create An EC2

```
{
  "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
  "Parameters": {
    "KeyPair": {
      "Description": "The EC2 Key Pair to allow SSH access to the instance",
      "Type": "String"
    }
  },
  "Resources": {
    "Ec2Instance": {
      "Properties": {
        "ImageId": "ami-9d23aeea",
        "InstanceType": "m3.medium",
        "KeyName": {
          "Ref": "KeyPair"
        }
      },
      "Type": "AWS::EC2::Instance"
    }
  },
  "Outputs": {
    "InstanceId": {
      "Description": "The InstanceId of the newly created EC2 instance",
      "Value": {
        "Ref": "Ec2Instance"
      }
    }
  },
  "AWSTemplateFormatVersion": "2010-09-09"
}
```

You will be asked to enter values for these parameters when you create your stack

Sample Template – Create An EC2

```
{
  "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
  "Parameters": {
    "KeyPair": {
      "Description": "The EC2 Key Pair to allow SSH access to the instance",
      "Type": "String"
    }
  },
  "Resources": {
    "Ec2Instance": {
      "Properties": {
        "ImageId": "ami-9d23aeea",
        "InstanceType": "m3.medium",
        "KeyName": {
          "Ref": "KeyPair"
        }
      },
      "Type": "AWS::EC2::Instance"
    }
  },
  "Outputs": {
    "InstanceId": {
      "Description": "The InstanceId of the newly created EC2 instance",
      "Value": {
        "Ref": "Ec2Instance"
      }
    }
  },
  "AWSTemplateFormatVersion": "2010-09-09"
}
```

Includes statically defined properties (ImageID & Instance Type) plus a reference to the KeyPair parameter. ImageID is the AMI specific to the region that you will launch this stack in, in this case the eu-west-1 region

Sample Template – Create An EC2

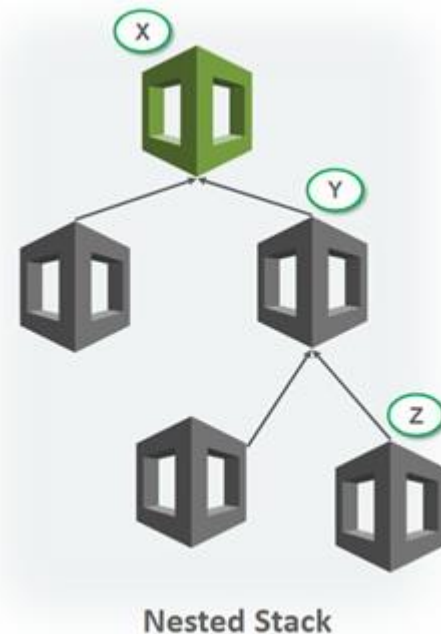
```
{
  "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
  "Parameters": {
    "KeyPair": {
      "Description": "The EC2 Key Pair to allow SSH access to the instance",
      "Type": "String"
    }
  },
  "Resources": {
    "Ec2Instance": {
      "Properties": {
        "ImageId": "ami-9d23aeea",
        "InstanceType": "m3.medium",
        "KeyName": {
          "Ref": "KeyPair"
        }
      },
      "Type": "AWS::EC2::Instance"
    }
  },
  "Outputs": {
    "InstanceId": {
      "Description": "The InstanceId of the newly created EC2 instance",
      "Value": {
        "Ref": "Ec2Instance"
      }
    }
  },
  "AWSTemplateFormatVersion": "2010-09-09"
}
```

These outputs will be returned once the template has completed execution

Stack In CloudFormation

Stack is a collection of AWS resources

- These resources are created, deleted or updated in stack and it is defined by AWS CloudFormation template
- A stack can be nested with another stack and this results into hierarchy of stacks
- The top-level stack is called as root stack, all other stacks in hierarchy belong to root stack
- Every nested stack has a parent stack and in case of first level root stack is the parent stack
- Example- X is the root stack for all plus parent stack for Y. Y is parent stack to Z





Elastic Beanstalk

Elastic Beanstalk

Elastic Beanstalk is a web hosting platform offered by Amazon

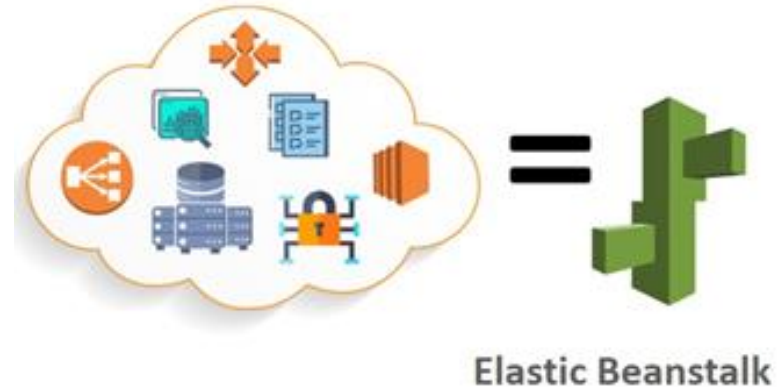
- Elastic Beanstalk is used as a PaaS for deploying and scaling web applications and services
- These web applications and services are developed via programming languages such as Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker web applications
- The servers like Apache, Tomcat, IIS and Nginx are used for deployment



Elastic Beanstalk

Why Do We Need Elastic Beanstalk

- When a developer deploys an application using Elastic Beanstalk, it automates services such as Load Balancing, Provisioning servers, Auto Scaling, creating environments to run Versions of applications, maintaining security, basic health checks and monitoring.
- The AWS Toolkit for Visual Studio and the AWS Toolkit for Eclipse allow you to deploy your application to AWS Elastic Beanstalk and manage it without leaving your IDE.
- Hence, developers can focus on developing their application and are set free from deployment-oriented tasks.



Features Of Elastic Beanstalk



Elastic Beanstalk

01

Elastic Beanstalk is the fastest and simplest way to deploy your application on AWS

02

Enables you to focus on writing code rather than spending time on managing and configuring services

03

Automatically scales your application up and down based on applications specific needs

04

Gives freedom to select AWS resources, like EC2 instance type that are optimal of your applications

Components Of Elastic Beanstalk

Application

App Versions

Environment

Environment Tier

Environment Health

- An *application* is collection of *environments*, *versions* and *environment configuration*
- An application in Elastic Beanstalk is conceptually similar to a *folder*
- *Example:* tomcat- webapp

Components Of Elastic Beanstalk

Application

App Versions

Environment

Environment Tier

Environment Health

- An Application Version refers to a specific labeled iteration of deployable code for a web application
- An App version points to an Amazon S3 object that contains the deployable code such as a Java WAR file

Components Of Elastic Beanstalk

Application

App Versions

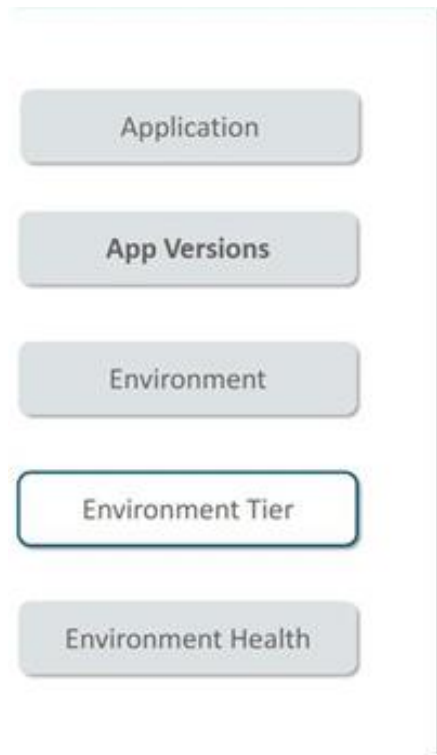
Environment

Environment Tier

Environment Health

- *Environment* is the place where you actually *launch* your *application*
- When you launch an Elastic Beanstalk environment, Beanstalk starts assigning various *AWS resources* to your *application*
- Each Environment runs only a *single application version* at a time
- *Example:* tomcat-webapp-dev, tomcat-webapp-prod





Components Of Elastic Beanstalk



- There are two types of tier: Web Server Tier and Worker Tier
- Web Server tier serves HTTP requests
- Worker tier looks after background-processing and background tasks

Components Of Elastic Beanstalk



- Elastic Beanstalk reports the **health** of a web server environment depending on how the application which is running in it responds to the **health check**
- It uses four colors to describe the status —
 -  Environment is being updated
 -  Passed recent health check
 -  Failed one or more checks
 -  Failed three or more checks



OpsWorks

AWS OpsWorks

AWS OpsWork is a configuration management service that provides managed instances of Chef and Puppet

- The automated platforms like Chef and Puppet permits the user to use code to automate their server configurations
- It makes easy to manage complete application Lifecycle including resource provisioning, configuration, management, application deployment, software updates, monitoring and access control



Why Do We Need OpsWorks?

- AWS OpsWorks is a configuration Management Service that helps you build and Operate highly dynamic applications and propagate changes instantly.

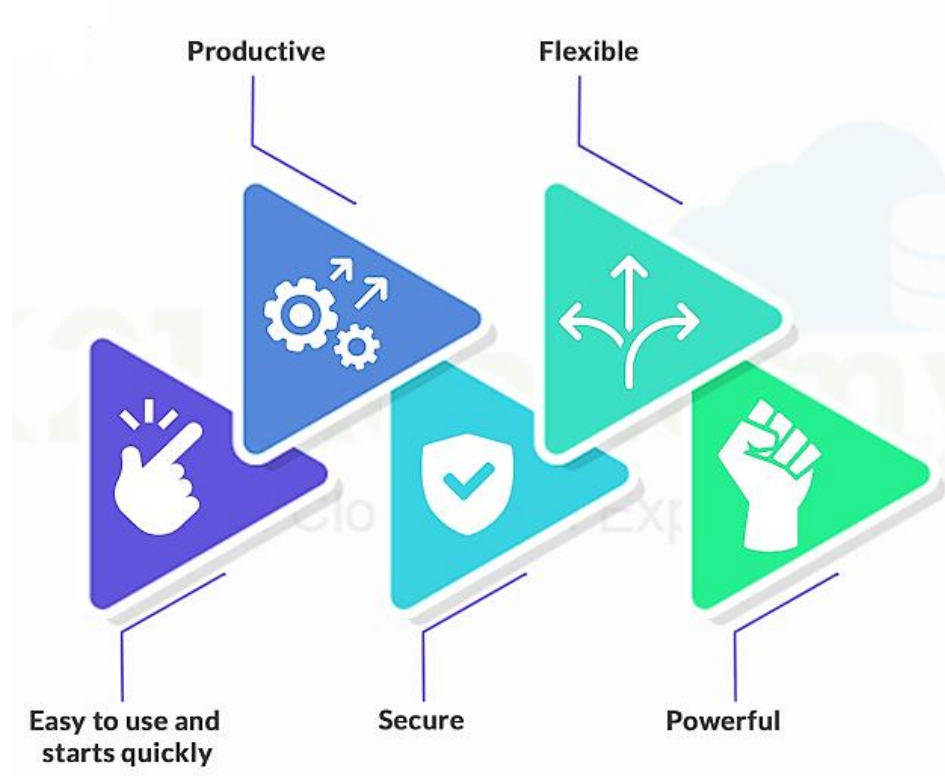
Suppose you want to change *properties file* of *1000 servers* running at a time

With the help of OpsWorks tools you can do it seamlessly

All you must do is only once deploy the changes and it will replicate them across all the components.

- The reasons why should we choose OpsWorks are-

Why Do We Need OpsWorks?

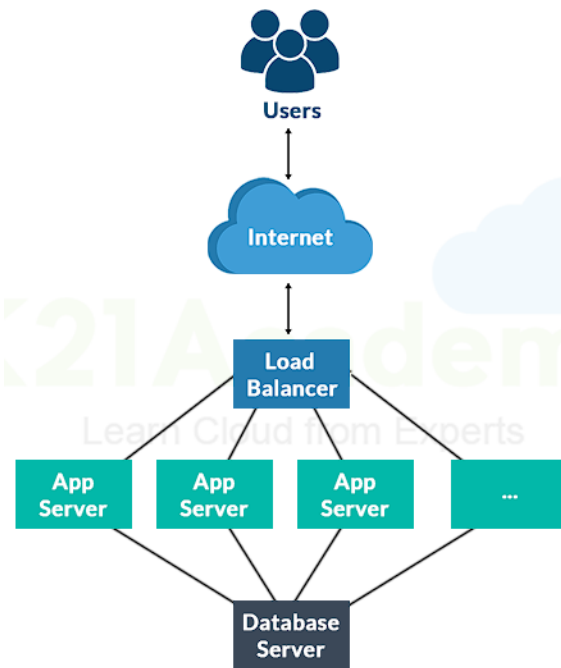




AWS OpsWorks Stacks

AWS OpsWorks Stacks

- AWS OpsWorks Stacks lets you manage applications and servers on AWS and on-premises
- Using it you can model your application as a stack containing different layers, such as load balancing, database, and application server
- The OpsWorks Stacks service helps you model, provision, and manage your applications on AWS using the embedded Chef solo client that is installed on Amazon EC2 Instances on your behalf
- It is mainly used by System administrators and ops-minded developers who are looking for a powerful end-to-end configuration management solution





Components Of OpsWorks Stacks

Components Of OpsWorks Stacks

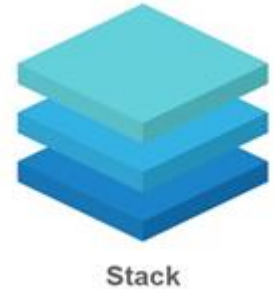
Stack

Layer

Instances

Apps

- Stack is top level of AWS OpsWorks entity
- It mainly represents set of resources that we want to manage collectively
Example-EC2 instances, EBS volumes, Load Balancers
- It handles tasks like managing applications and cookbooks, to the group of instances



Components Of OpsWorks Stacks

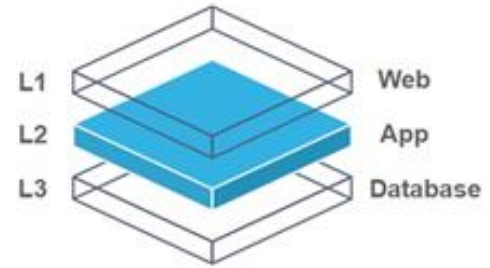
Stack

Layer

Instances

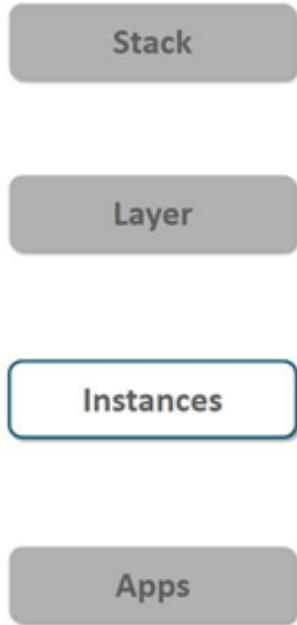
Apps

- Every stack contains one or more layers.
- Each layer represents stack components, like a load balancer or a set of application servers
- Each layer and a stack must have at least one instance and can optionally have multiple instances



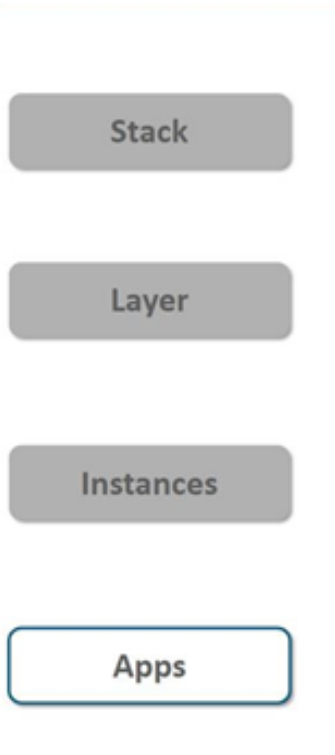
Example: DEV Stack

Components Of OpsWorks Stacks



- An instance represents a computing resource, like Amazon EC2 instance, which handles the work of serving applications
- Types of Instances Supported in AWS OpsWorks
 - 24/7 Instances:** You must start these instances and they keep running until you stop them
 - Time-Based Instances:** Instances run by AWS Ops Works on a specified daily and weekly schedule. Here, stack automatically adjust the number of instances to accommodate predictable usage patterns
 - Load-Based Instances:** These instances are turned on and off by OpsWorks based on specified load metrics such as CPU utilization

Components Of OpsWorks Stacks



- An AWS OpsWorks app represents code that you want to run on an application server.
- The app contains the information required to deploy the code to the appropriate application server instances.
- You can deploy apps in the following ways:
 - **Automatically:** When you start instances, AWS Ops Works automatically runs the instance's deploy recipes .
 - **Manually:** If you have a new app or want to update an existing one, you can manually run the online instances deploy recipes.





AWS OpsWorks Services

AWS OpsWorks Services



AWS OpsWorks for Chef Automate and AWS OpsWorks Stacks let you use Chef cookbooks and solutions for configuration management, while OpsWorks Puppet Enterprise lets you configure a Puppet Enterprise master server in AWS

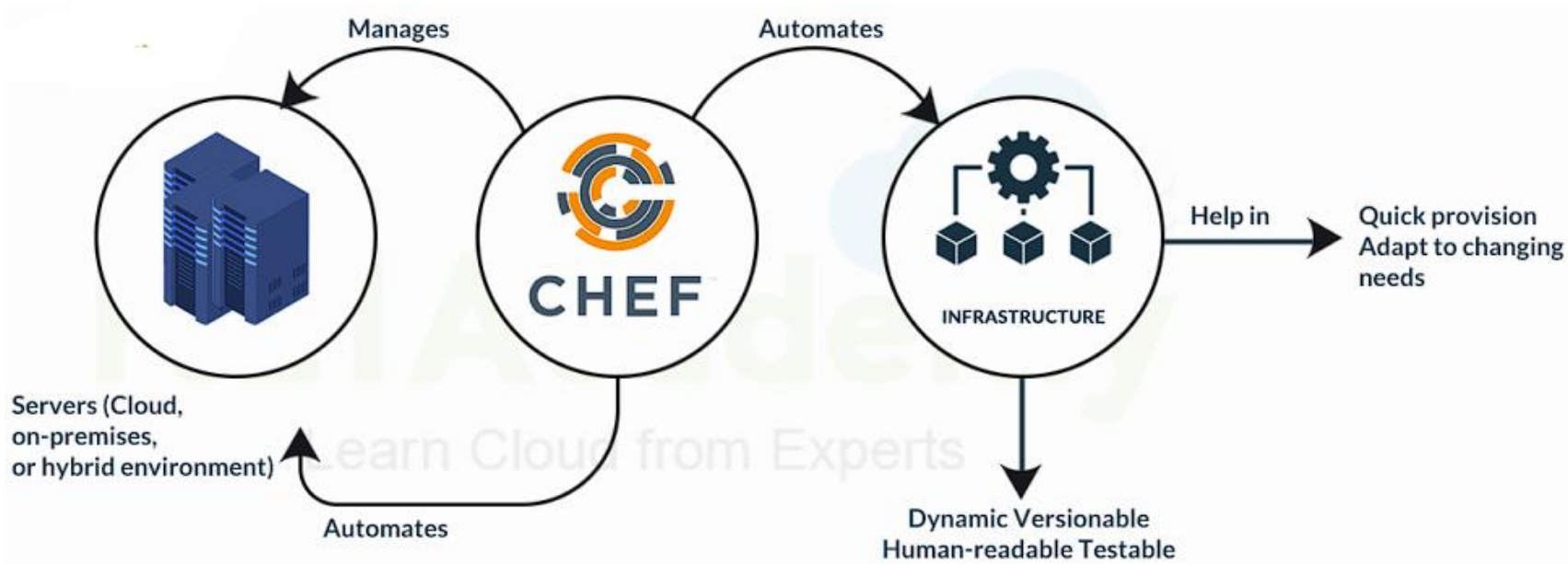


AWS OpsWorks For Chef Automate

AWS OpsWorks For Chef Automate

- AWS Ops Works for Chef is a configuration management service which provides Chef server and lets the service operate it along with backups and software updates
- It is compatible with Chef's Supermarket cookbooks and recipes plus supports the tools like knife and test kitchen
- It is mainly used by people looking for a configuration management experience i.e., fully compatible with Chef including all community scripts and tooling
- It helps you in automating entire Infrastructure and managing them across multiple end points

AWS OpsWorks For Chef Automate



Cookbook

- A cookbook is like a package” for Chef recipes
- It contains all the recipes, files, templates, libraries, etc. required to configure a portion of your Infrastructure

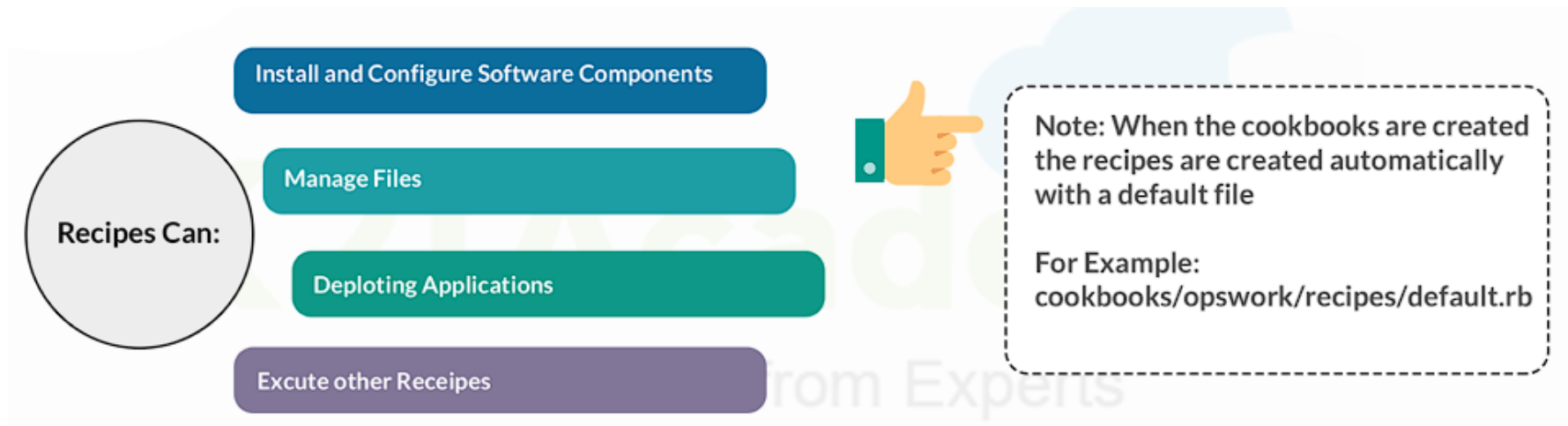
To create a cookbook

```
Knife cookbook create opswork(name of cookbook)
```

// The cookbook created is a directory format where in all the subfolders like recipes etc. get created automatically

Recipes

- Recipes are the codes written in Ruby language and attached to the layer.
- They are also known as configuration files that describe resources(building blocks of chef) and their desired state



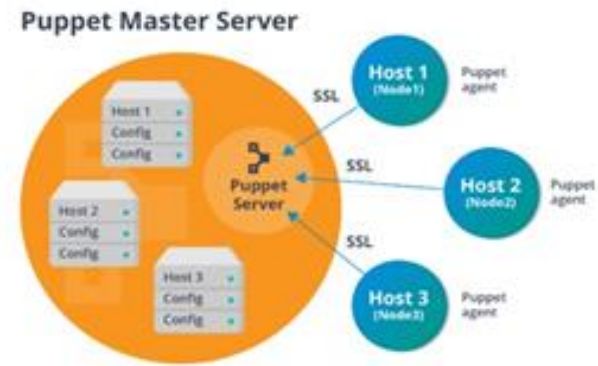


AWS OpsWorks For Puppet Enterprise

AWS OpsWorks For Puppet Enterprise

OpsWork for Puppet Enterprise provides managed puppet enterprise servers with set of automation tools which offers **workflow** automation for **orchestration**, **automated provisioning** and **visualization**

- It lets you to create AWS managed master servers
- Master server manages nodes in infrastructure, holds information of those nodes and acts as central repository for puppet modules
- Modules are the reusable and sharable units of puppet code, containing instructions about infrastructure configuration



AWS OpsWorks For Puppet Enterprise

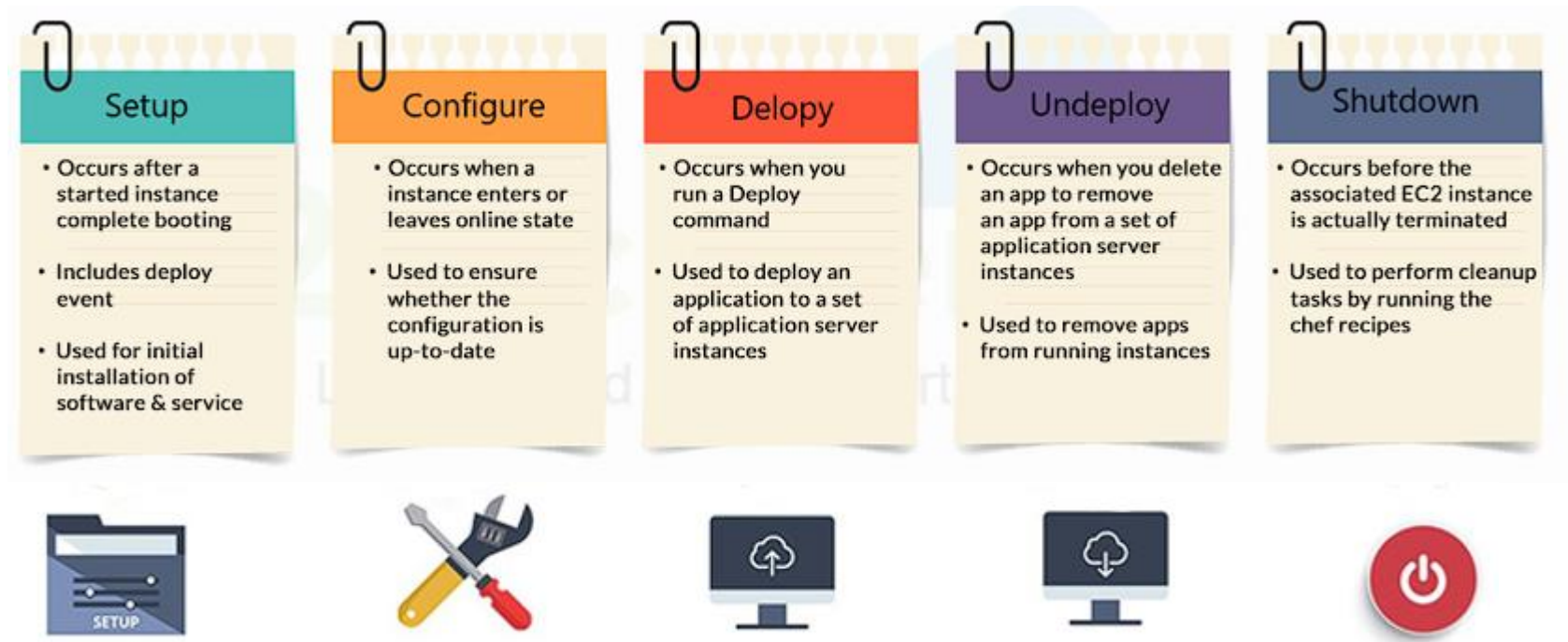
OpsWork for Puppet Enterprise provides managed puppet enterprise servers with set of automation tools which offers *workflow* automation for *orchestration*, *automated provisioning* and *visualization*

- Its serve to provision full stack automation for managing tasks like software and Os's configuration, package installation, database setup and more
- It is mainly used by people In search of managed configuration management experience i.e., compatible with puppet along with all puppet forge modules for puppet enterprise without operational over head



AWS OpsWorks Lifecycle Events

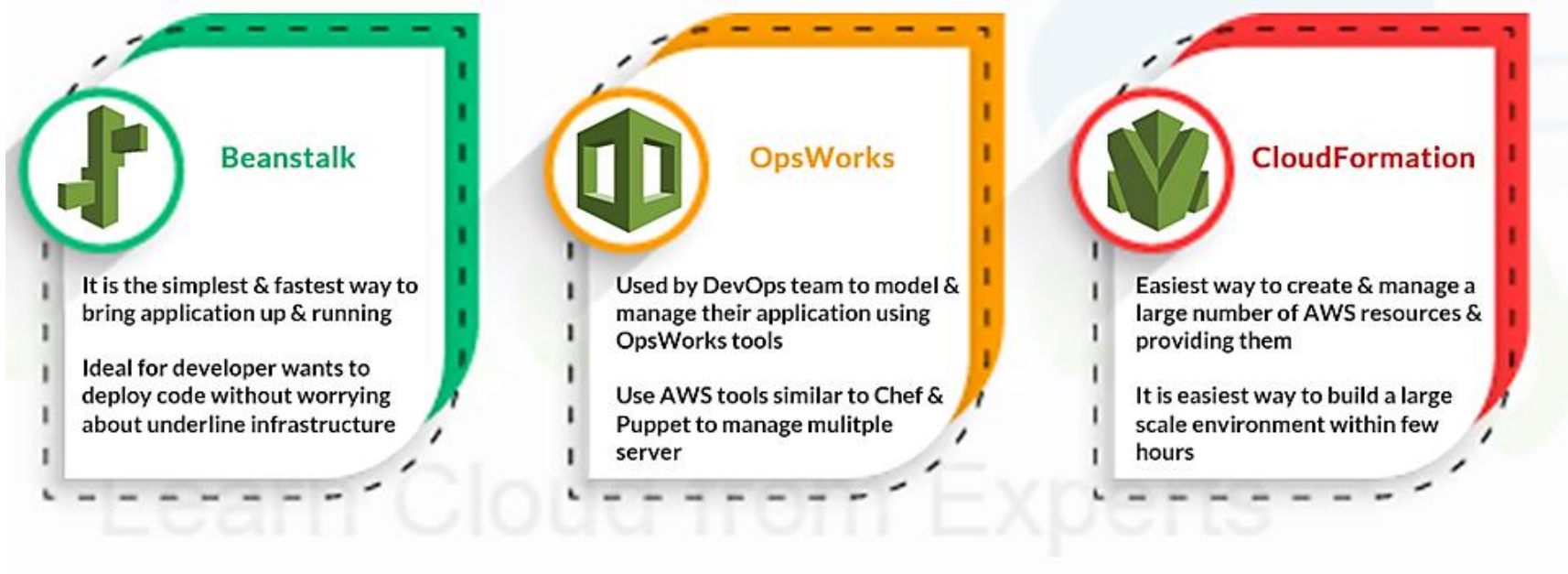
OpsWorks Lifecycle Event





Beanstalk V/S OpsWorks V/S CloudFormation

Beanstalk v/s OpsWorks v/s CloudFormation



Lab Exercises

Creating and Updating Stacks using CloudFormation

- Creating Cloudformation stacks
- Updating Cloudformation stacks



Contents

1	Introduction	3
2	Documentation Links	5
3	Creating and updating cloudformation stacks	6
3.1	Creating an EC2 instance using CloudFormation	6
3.2	Updating the CloudFormation Stack	14
3.3	Creating and deploying a stack for User-data	23
3.3.1	Creating a Key Pair	23
3.3.2	Creating and deploying the stack	24
4	Delete/cleanup	34
4.1	Deleting the Updated Stack	34
4.2	Deleting the User-data Stack	36
5	Troubleshooting	38
5.1	Failed to create Stack	38
6	Summary	39

Lab Exercises

Deploy Web Application Using Elastic Beanstalk

- Deploying a website using AWS Elastic Beanstalk



Contents

1	Introduction	3
2	Documentation Links	4
3	Deploying a Website Using AWS Elastic Beanstalk	5
3.1	Creating an Application in AWS Elastic Beanstalk.....	5
3.2	Resources Created by Elastic Beanstalk in Background.....	10
3.3	Mapping the Domain Name of Elastic Beanstalk Using Route53	12
4	Deleting/Cleanup	16
4.1	Terminate the Environment and then Delete the Application.....	16
4.2	Delete the Route53 Hosted Zone.....	19
5	Summary.....	21

Quiz Question

You are tasked with reducing cost in an organization that does several new application deployments every week. The company has a common stack of resources but is spending time each deployment re-creating this stack. What would help you reduce initial setup and deployment costs?

- A. Elastic Beanstalk
- B. CloudFormation
- C. AWS Trusted Advisor.
- D. Application load balancers

Answer B

Explanation: CloudFormation allows you to automate provisioning and, in this case, to create standardized JSON scripts that can be lightly modified to stand up entire stacks for multiple applications that share a common structure.

Quiz Question

Your company is paying a high cost for a consultant whose only job is the provisioning of resources for new cloud deployments. What AWS service would allow you to reduce this expenditure and move the consulting into more of a business-serving capacity?

- A. Elastic Beanstalk
- B. CloudTrail
- C. CloudShift
- D. CloudFormation

Answer : A

Explanation: CloudFormation is ideal for automating deployment without manual intervention, but it's actually Elastic Beanstalk that handles the *provisioning* of resources.

Find Us



<https://www.facebook.com/K21Academy>



<http://twitter.com/k21Academy>



<https://www.linkedin.com/company/k21academy>



<https://www.youtube.com/k21academy>