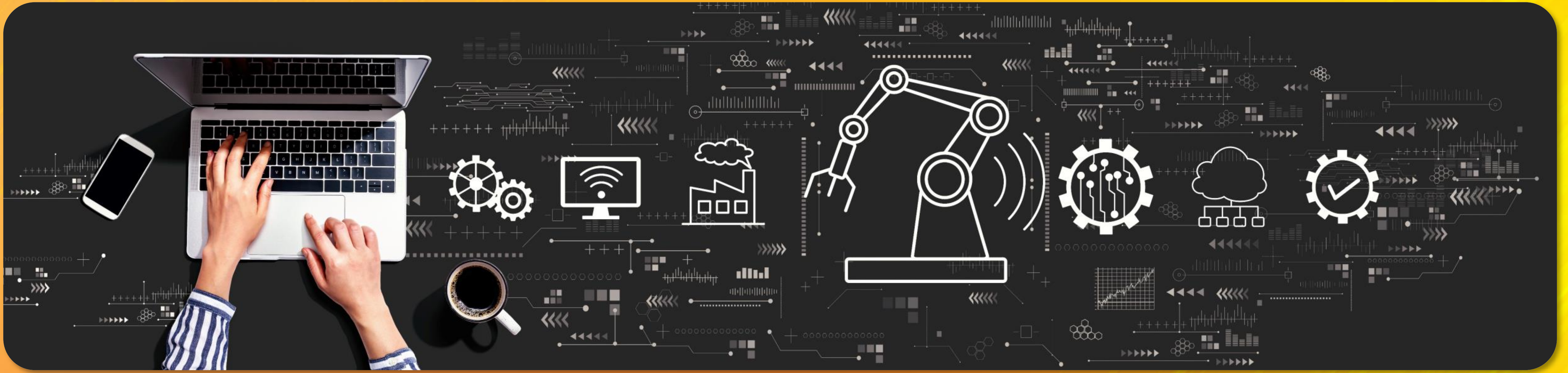
Automation in Cloud

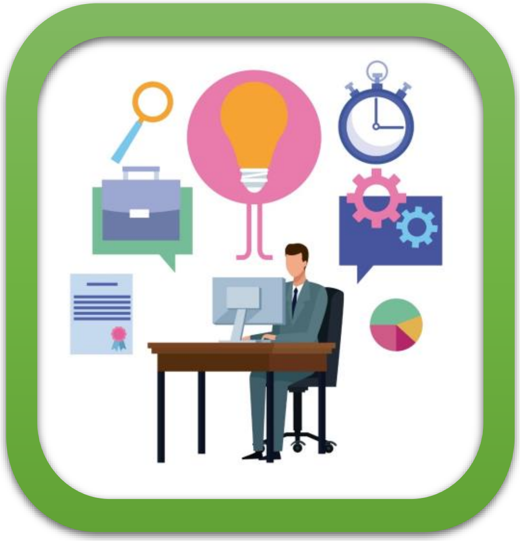


Agenda

- Why Automate?
- Infrastructure Automation
- Application Automation

Why Automate?

Why automate?

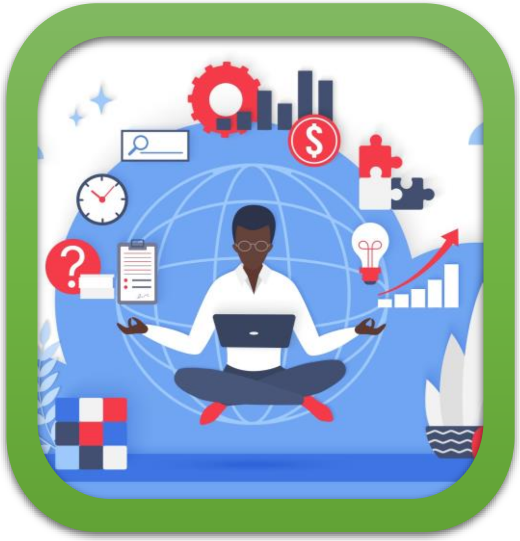


Efficiency

Productivity

Consistency

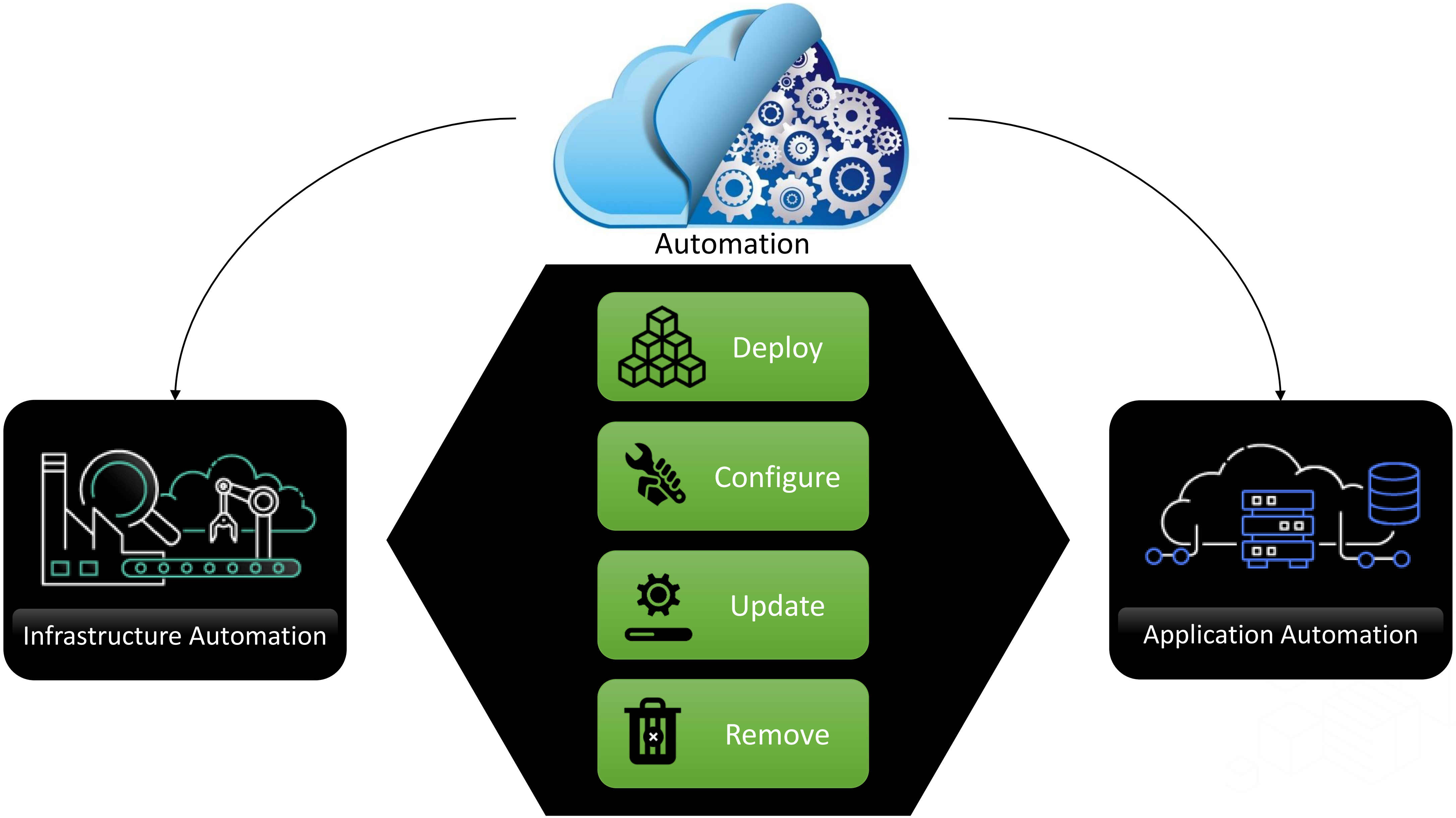
Quality



Repeatability

Recoverability

Automation in Cloud



Automation

Deploy

Configure

Update

Remove

Infrastructure Automation

Application Automation

Infrastructure Automation

Infrastructure Automation

- IT infrastructure are the components required to operate and manage enterprise IT environments. These components include hardware, software, networking components, operating systems, and storage—all of which are used to deliver IT services and solutions.

- Infrastructure automation enables you to automate manual processes and speed up the delivery of infrastructure resources on a self-service basis, according to user or business demand.

- It reduces human interaction with IT systems by creating scripts or functions that are repeatable and can be used either by other software or on command.

AWS CloudFormation

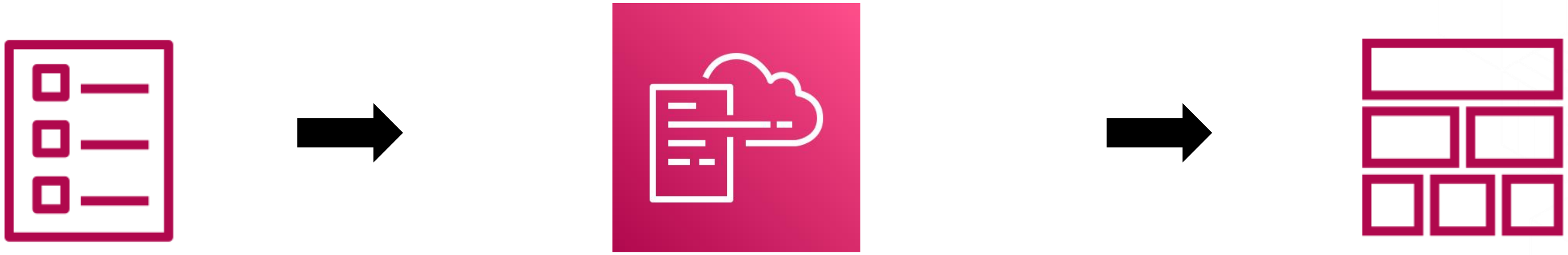
Constructing an apartment

Bedroom
Cabinet
Kitchen
Living
Bedroom
Bedroom
Bathroom

Blueprint

Builder

Apartment



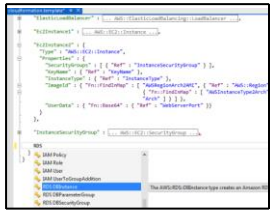
Template

AWS CloudFormation

Stack

How CloudFormation works?

Create Manually



```
cloudformation.template*  ✕
    "ElasticLoadBalancer" : {... AWS::ElasticLoadBalancing::LoadBalancer ...},

    "Ec2Instance1" : {... AWS::EC2::Instance ...},

    "Ec2Instance2" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "InstanceType" : { "Ref" : "InstanceType" },
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region"
                                        { "Fn::FindInMap" : [ "AWSInstanceType2Arch"
                                        "Arch" ] } ] },
        "UserData" : { "Fn::Base64" : { "Ref" : "WebServerPort" }}
      }
    },

    "InstanceSecurityGroup" : {... AWS::EC2::SecurityGroup ...},
    RDS
  }
}
```

The AWS::RDS::DBInstance type creates an Amazon RD

Sample Templates

Templates & Snippets by AWS Service

Browse sample templates by AWS service.

Browse template snippets by AWS service.

Refer to our developer documentation for more examples and reference.

Browse sample templates by AWS region.

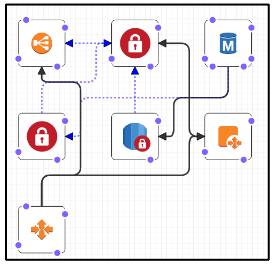
Application Frameworks

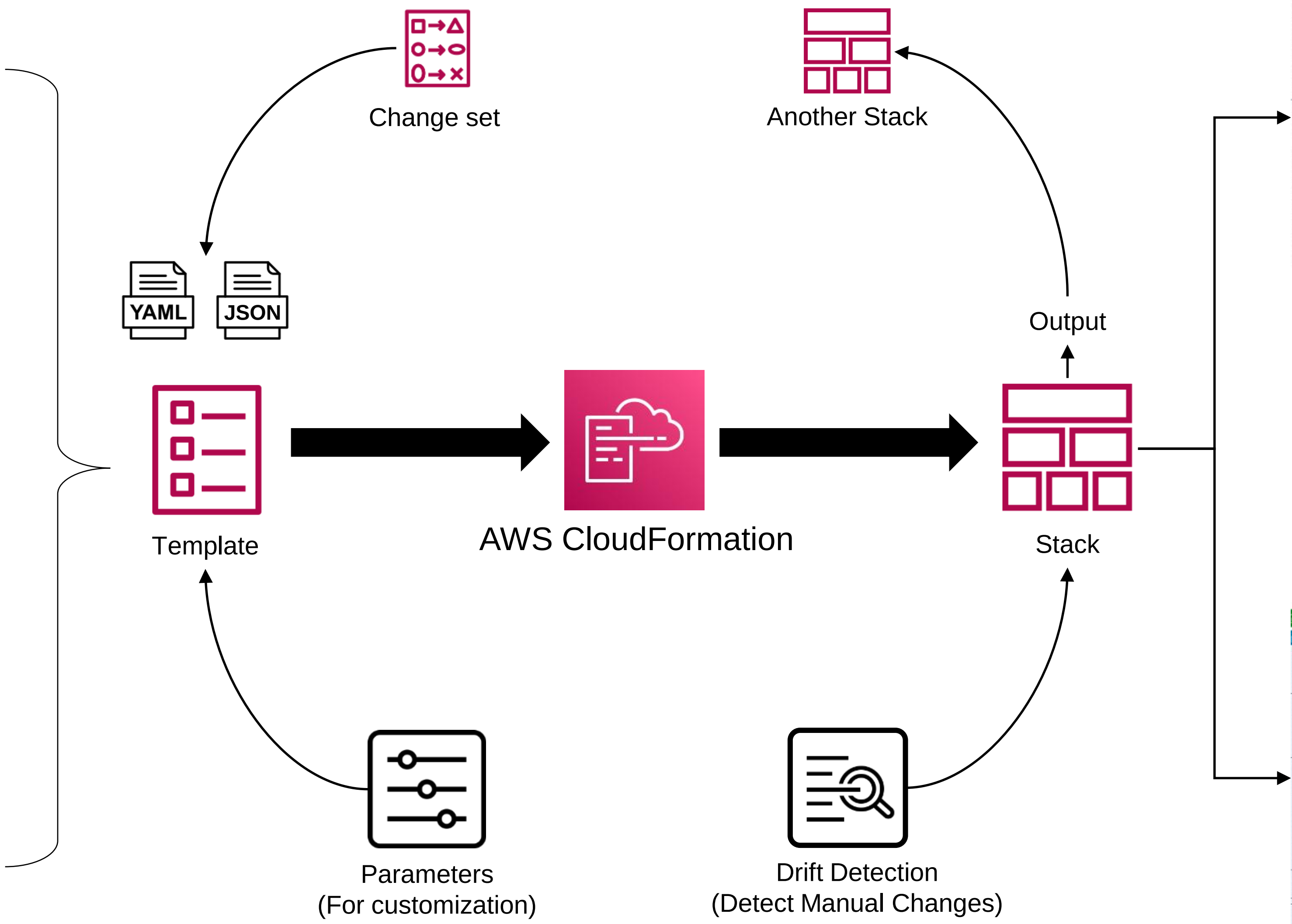
Application framework templates demonstrate how to use AWS CloudFormation to provision popular frameworks such as LAMP and Ruby on Rails.

Experimental and Community Samples

Browse AWS Labs for experimental AWS CloudFormation templates and other projects contributed by AWS employees and partners.

CloudFormation Designer





Change set

Another Stack

YAML

JSON

Output

Template

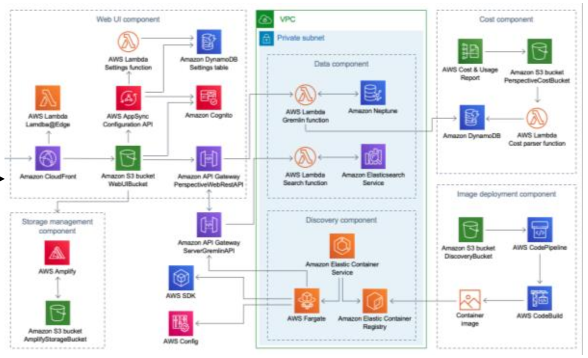
AWS CloudFormation

Stack

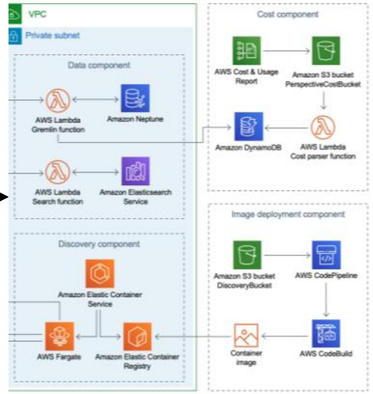
Parameters
(For customization)

Drift Detection
(Detect Manual Changes)

Production Environment

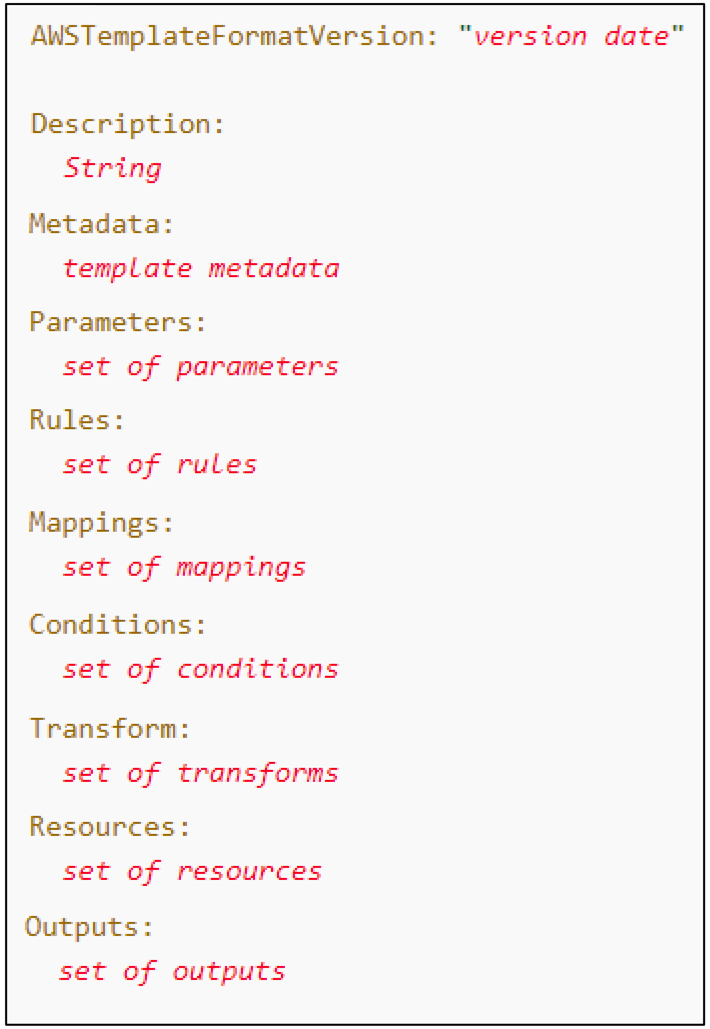


QA Environment



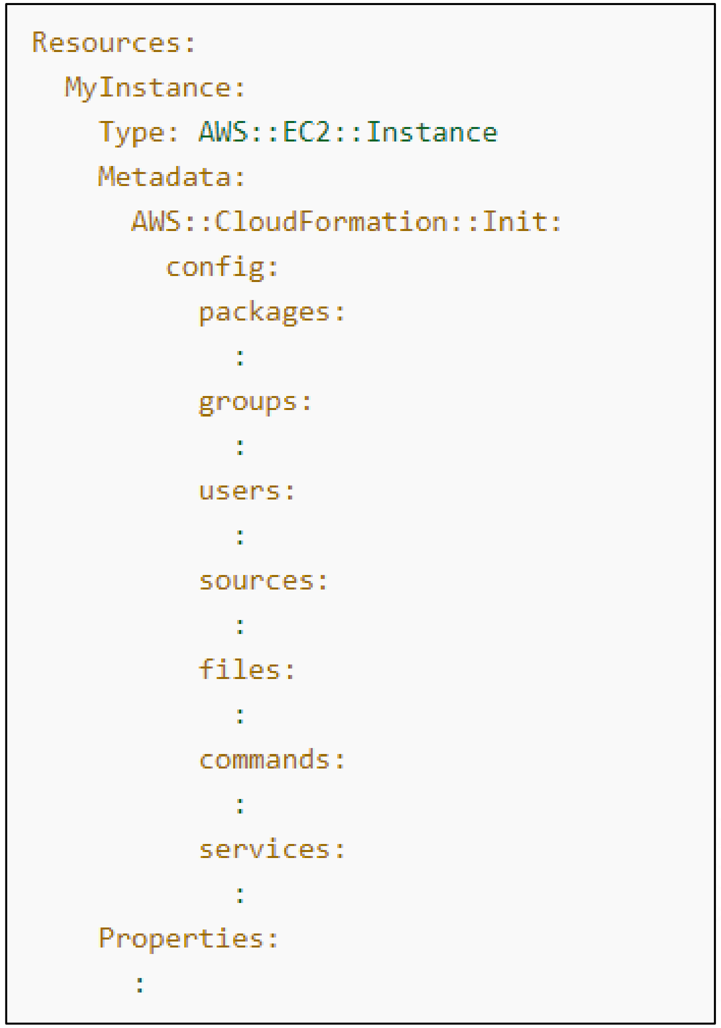
CloudFormation Template

Configuring the Infrastructure (Infrastructure Automation)



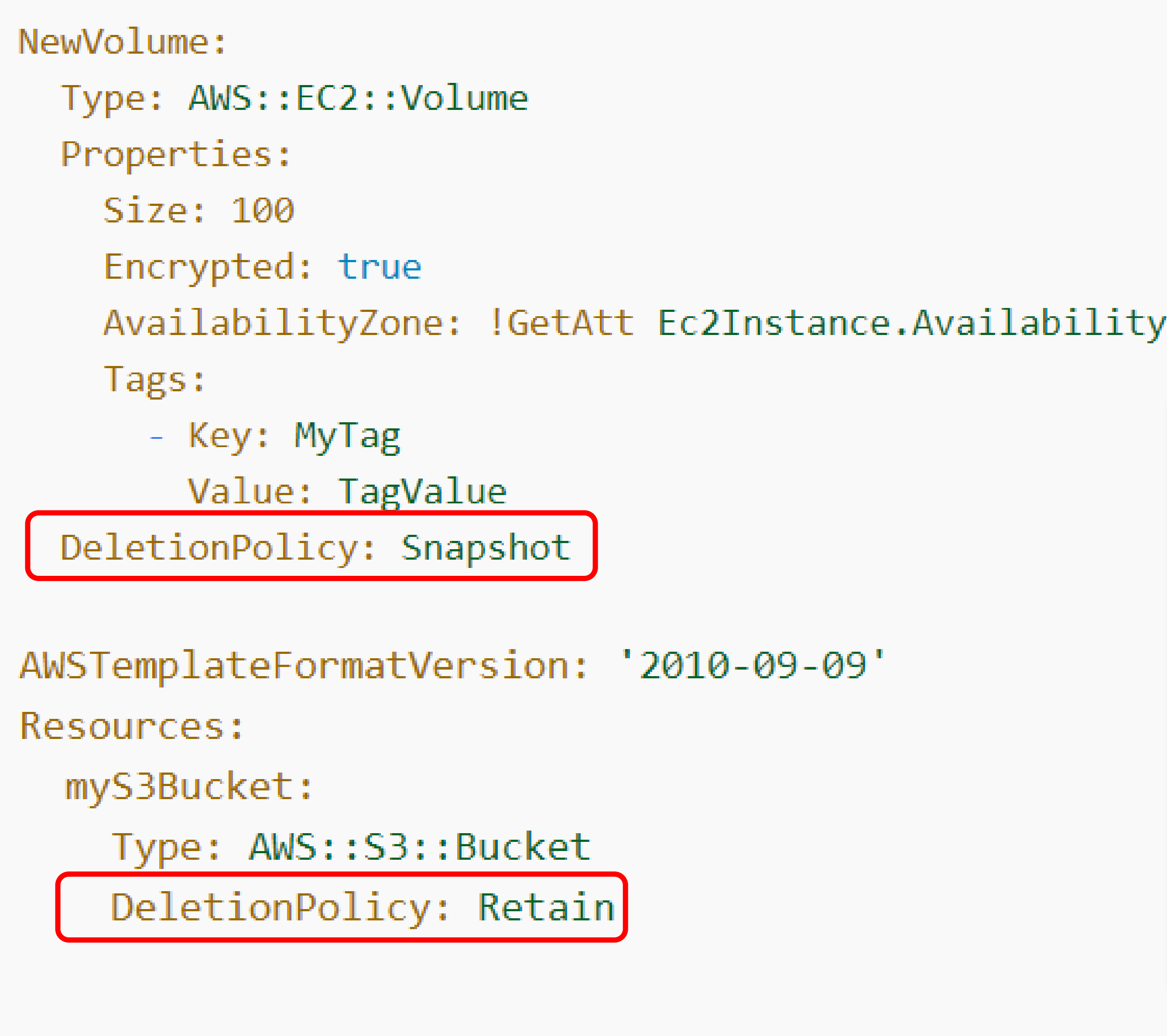
```
AWSTemplateFormatVersion: "version date"

Description:
  String
Metadata:
  template metadata
Parameters:
  set of parameters
Rules:
  set of rules
Mappings:
  set of mappings
Conditions:
  set of conditions
Transform:
  set of transforms
Resources:
  set of resources
Outputs:
  set of outputs
```

Configuring the Instance (Application Automation)



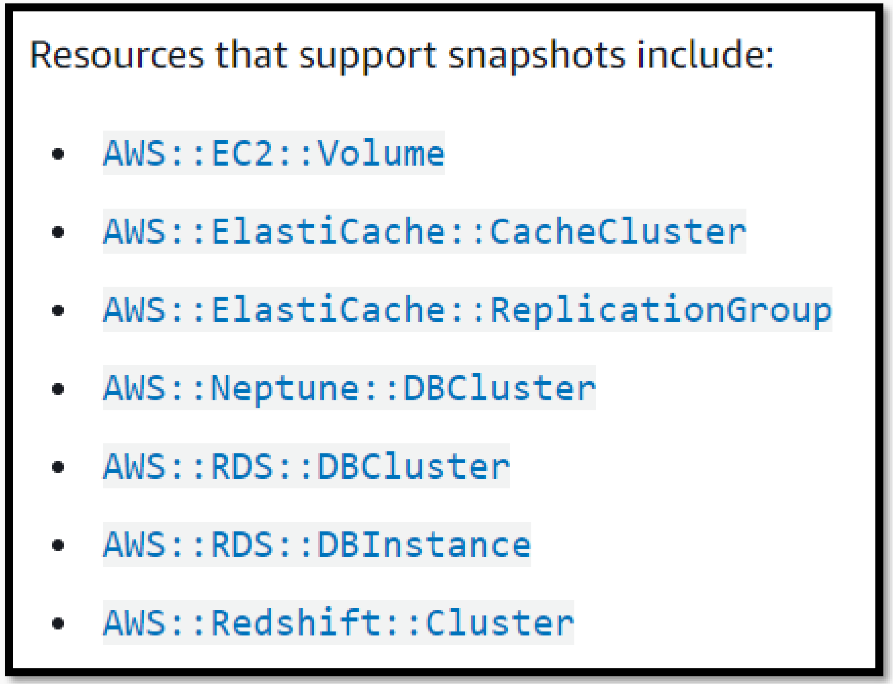
```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            :
          groups:
            :
          users:
            :
          sources:
            :
          files:
            :
          commands:
            :
          services:
            :
    Properties:
      :
```

Deletion Policy



```yaml
NewVolume:
  Type: AWS::EC2::Volume
  Properties:
    Size: 100
    Encrypted: true
    AvailabilityZone: !GetAtt Ec2Instance.AvailabilityZone
    Tags:
      - Key: MyTag
        Value: TagValue
  DeletionPolicy: Snapshot
```

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myS3Bucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
```



Resources that support snapshots include:

- `AWS::EC2::Volume`
- `AWS::ElastiCache::CacheCluster`
- `AWS::ElastiCache::ReplicationGroup`
- `AWS::Neptune::DBCluster`
- `AWS::RDS::DBCluster`
- `AWS::RDS::DBInstance`
- `AWS::Redshift::Cluster`

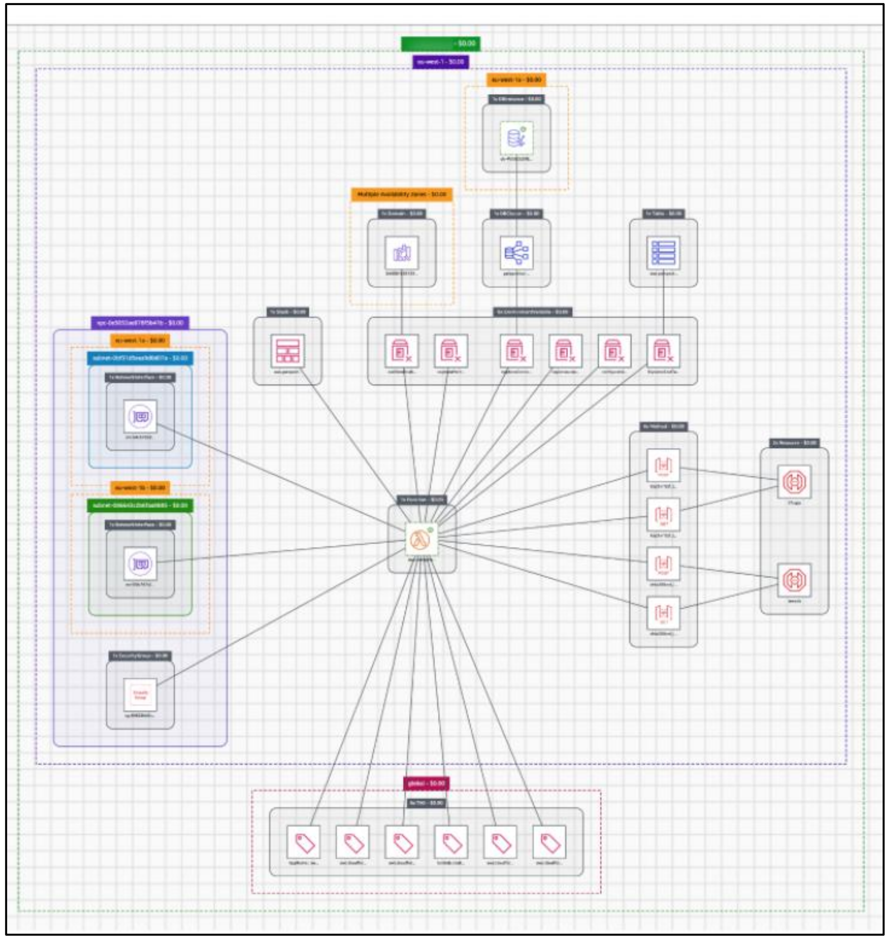
Other Popular Automation Products

- Terraform
  - CloudFormation Alternate

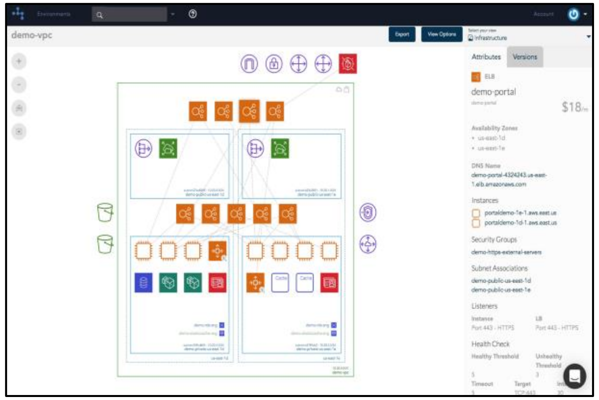
- Former2
  - Reverse Engineering - Create a template from existing resources

Architecture Diagram of your AWS Resources

- AWS Perspective (Solution)



- 3^{rd} Party Solutions
  - Hava.io



demo-portal

$18/m

Availability Zones
- us-east-1d
- us-east-1e

DNS Name
demo-portal-4324243.us-east-1.elb.amazonaws.com

Instances
- portaldemo-1e-1.aws.east.us
- portaldemo-1d-1.aws.east.us

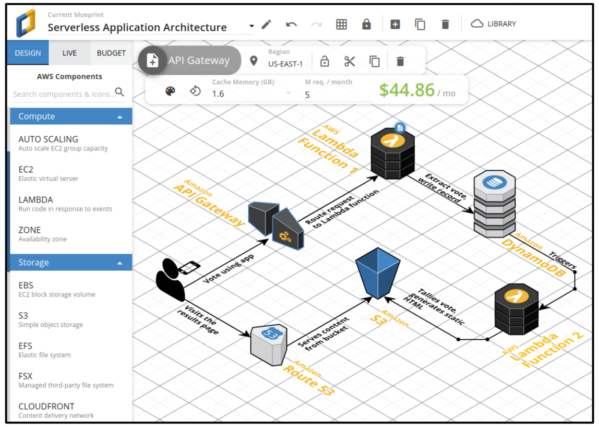
Security Groups
demo-https-external-servers

Subnet Associations
demo-public-us-east-1d
demo-public-us-east-1e

Listeners
Instance: Port 443 - HTTPS
LB: Port 443 - HTTPS

Health Check
Healthy Threshold: 5
Unhealthy Threshold: 3
Timeout: 5
Target: TCP:443

  - Cloudcraft.co



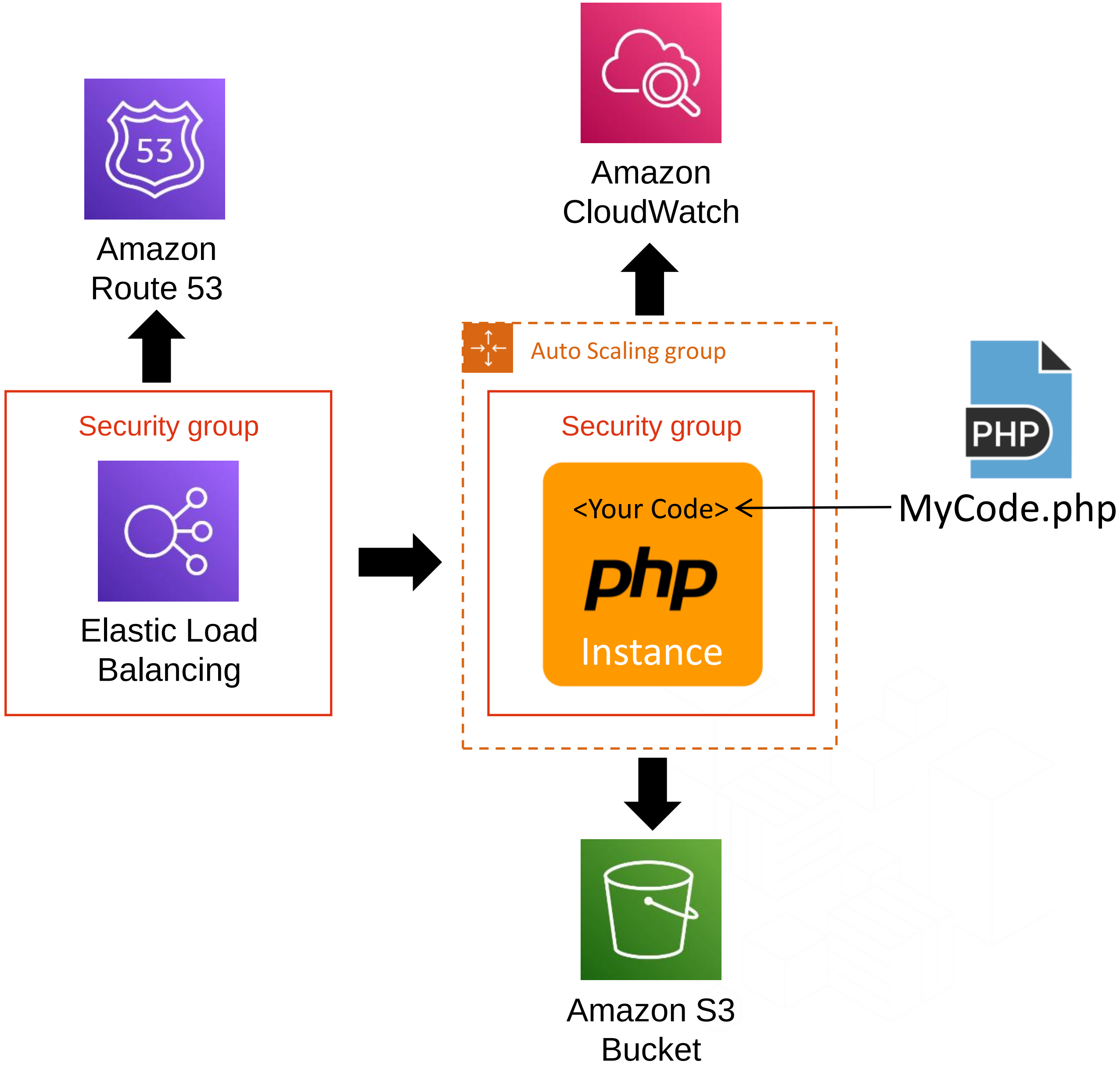
Cache Memory (GB): 1.6
M req. / month: 5
$44.86 / mo

AWS Elastic Beanstalk

Typical Application Deployment

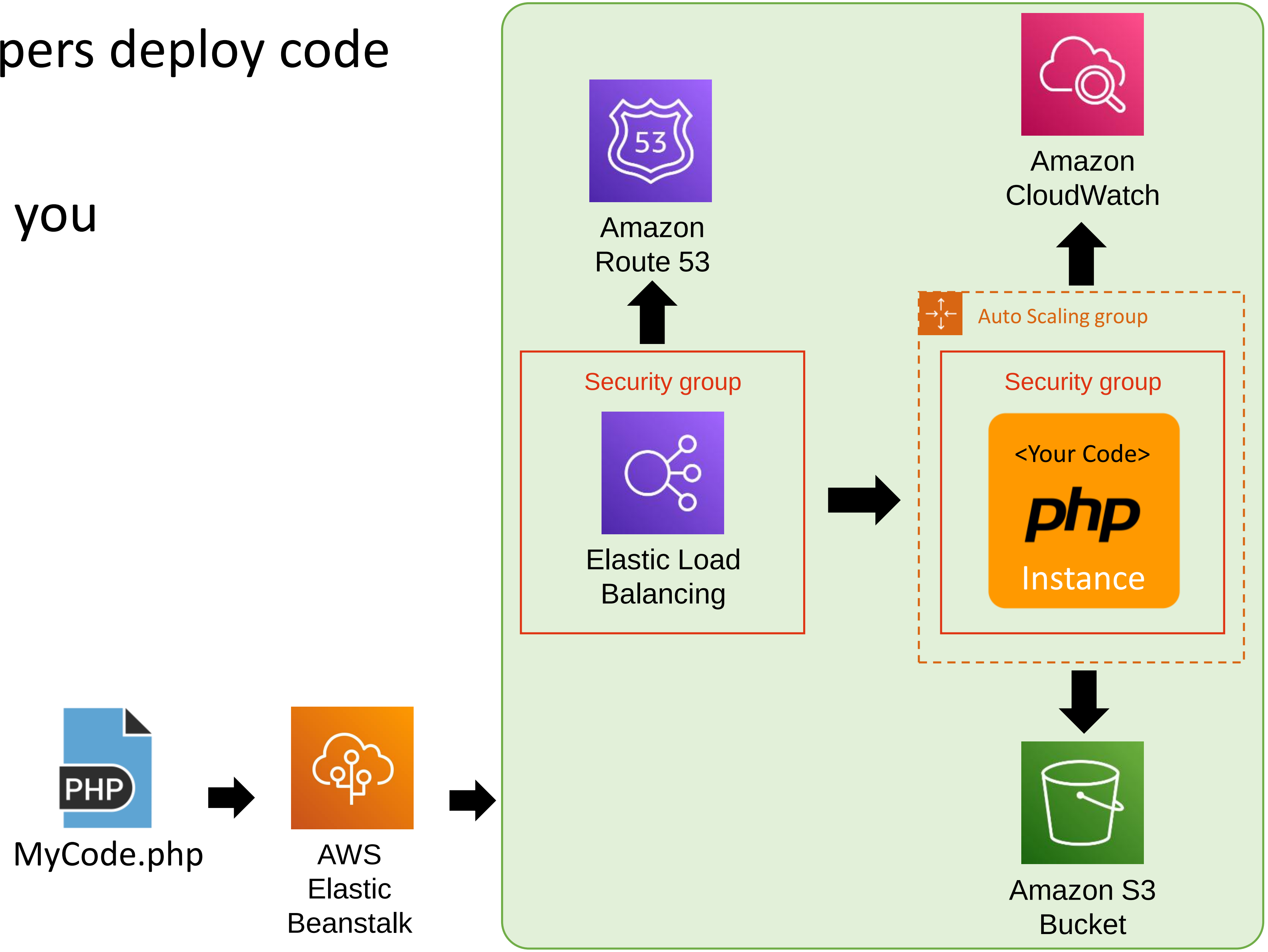
- You want to deploy your Code

1. Create EC2
2. Configure Security Groups
3. Install the supported Runtime
4. Configure Load Balancer
5. Configured Load Balancer Security Group
6. Configure AutoScaling
7. Create S3 Bucket for Logs
8. Configure CloudWatch
9. Register Domain
10. Deploy your Code
11. Plan for update
12. Rollout New version
13. Rollback if it didn’t worked as expected
14. Maintain a QA Environment



Application Deployment using AWS Elastic Beanstalk

- Designed to let developers deploy code
- Handles the details for you
- Compatible with:
  - PHP
  - Java
  - Python
  - Ruby
  - Node.js
  - .NET
  - Go
  - Docker applications



AWS OpsWorks

Configuration Management

- Configuration management is a process for maintaining computer systems, servers, and software in a desired, consistent state.

Current State	
Ubuntu OS	*19.04*
Java Version	*SE 16*
Service XYZ	*Enabled*
Open Port(s)	*80, 443*
Apache Tomcat	*Not Installed*

→ Configuration Management →

Desired State	
Ubuntu OS	20.04
Java Version	SE 18
Service XYZ	Disabled
Open Port	443
Apache Tomcat	3.3 Installed

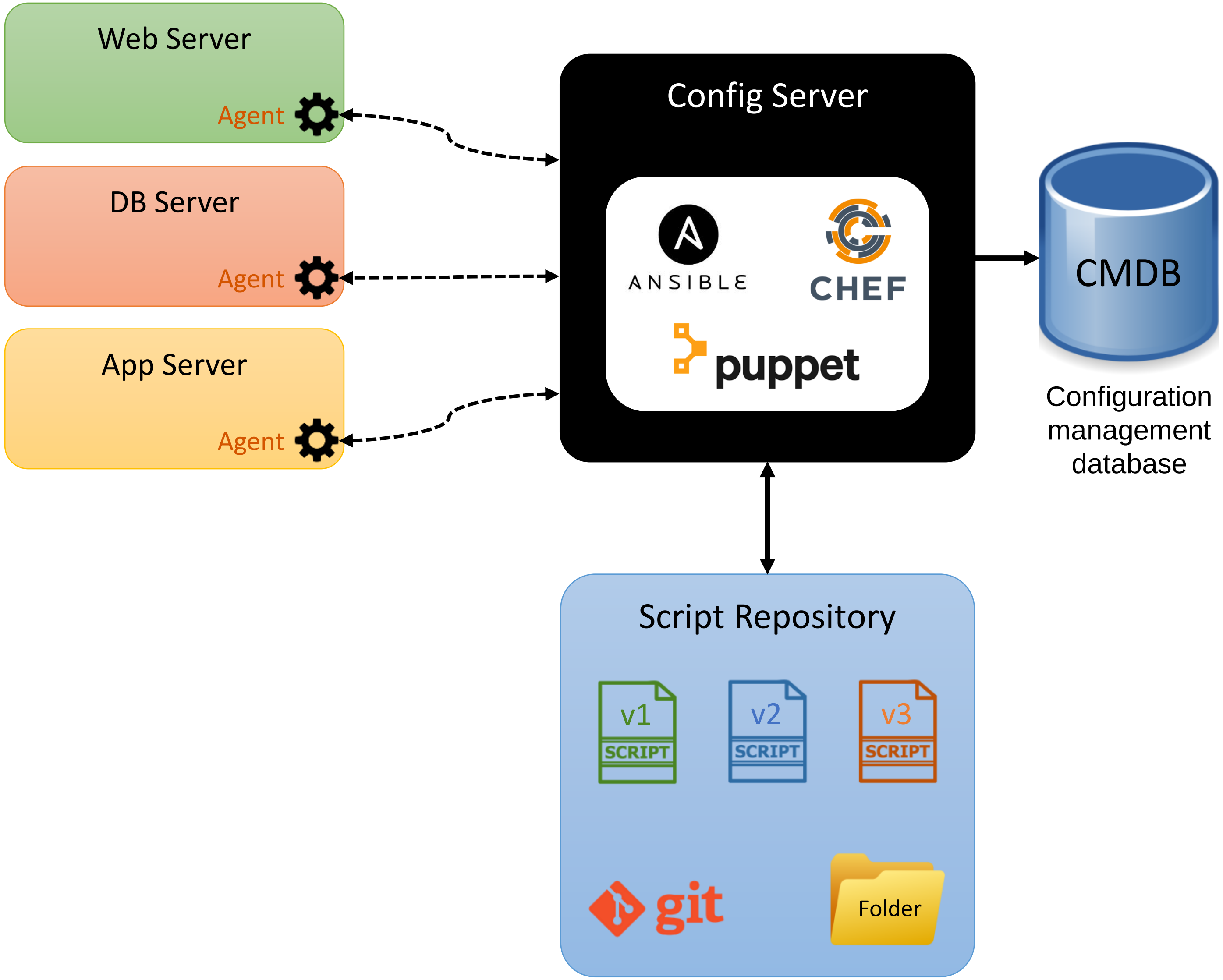
- Common configuration management tools

puppet

CHEF

ANSIBLE

How configuration management tools work?

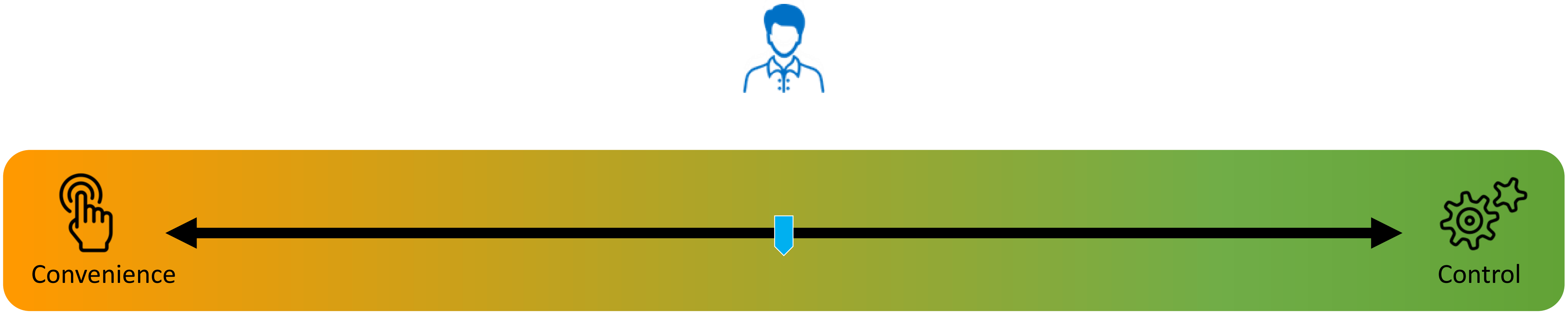


- Configuration Language
  - Ansible
    - YAML, Python
  - Puppet
    - Puppet DSL, Embedded Ruby
  - Chef
    - Ruby DSL, JSON
- Deployment Method
  - Push – Ansible
  - Pull – Puppet, Chef
- Scripts
  - Ansible – Playbook / Runbook
  - Puppet – Manifest / Catalog
  - Chef – Recipe / Cookbook

AWS OpsWorks

- AWS Opsworks for Chef Automate
  - A managed configuration management service that hosts Chef Automate
- AWS OpsWorks for Puppet Enterprise
  - A managed configuration management service that hosts Puppet master server
- AWS OpsWorks Stacks
  - Lets you manage applications and servers on AWS and on-premises
  - With OpsWorks Stacks, you can model your application as a stack containing different layers, such as load balancing, database, and application server

Choosing the right solution



Higher Level Services

AWS Elastic Beanstalk

AWS OpsWorks

Do it yourself

AWS CloudFormation

Amazon EC2

Reference:

Overview

Category:

Management and Governance

AWS OpsWorks

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet.
- Chef and Puppet are automation platforms that allow you to use code to automate the configurations of your servers.

Why?

- AWS OpsWorks frees you to focus on core configuration management tasks, instead of managing a Puppet master or Chef server. It eliminates the need for maintaining its infrastructure.
- It has three offerings, AWS Opsworks for Chef Automate, AWS OpsWorks for Puppet Enterprise, and AWS OpsWorks Stacks.

When?

- Choose AWS OpsWorks for Chef Automate if you are an existing Chef user. Choose AWS OpsWorks for Puppet Enterprise if you are an existing Puppet user. Choose AWS OpsWorks Stacks if you need a solution for application modeling and management using the abstractions of “stacks” and “layers”.

Where?

- AWS OpsWorks is a regional service.
- With AWS OpsWorks, you can automate how nodes are configured, deployed, and managed, whether they are Amazon EC2 instances or on-premises.

Who?

- AWS OpsWorks is a fully managed service.
- AWS creates a fully managed instance of Chef or Puppet running on Amazon EC2 and handles its operations, backups, restorations, and software upgrades.

How?

- You can get started by creating a Chef or Puppet server by using the AWS Management console, or the AWS CLI. Afterwards, you can import your existing assets (recipes, cookbooks, runbooks) or create new. You start an AWS OpsWorks Stacks project by creating a stack, which acts as a container for your instances and other resources.

How much?

- You pay for the number of nodes connected to Puppet master or Chef server, an hourly rate for the time those nodes are running, and for the underlying EC2 instance running your Puppet master or Chef server. There is no additional charge for OpsWorks Stacks on Amazon EC2, on-premises server running OpsWorks Stacks agent is charged hourly.

Reference:

FAQs

Category:

Management
and
Governance

AWS CloudFormation

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- AWS CloudFormation gives you an easy way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code (IaC).
- It enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).

Why?

- Automate, test, and deploy infrastructure templates with continuous integration and delivery (CI/CD) automations.
- Run anything from a single Amazon Elastic Compute Cloud (EC2) instance to a complex multi-region application.

When?

- You want to use a declarative way to create, update, and delete an entire stack as a single unit, instead of managing resources individually across multiple accounts and regions.
- You want predictable, controlled approach for managing resources across your application portfolio.

Where?

- AWS CloudFormation is a regional service, but it can deploy stacks across multiple accounts and regions using StackSets.
- A template is stored in an Amazon S3 bucket.

Who?

- You create or provide a template (JSON or YAML formatted text file) that describes all the AWS resources that you need, and CloudFormation takes care of provisioning and configuring those resources (stack) for you.
- When creating a stack, AWS CloudFormation makes underlying service calls to AWS to provision and configure resources.

How?

- You create a template (JSON or YAML formatted text file) that describes all the AWS resources that you, and CloudFormation takes care of provisioning and configuring those resources (stack) for you.
- When creating a stack, AWS CloudFormation makes underlying service calls to AWS to provision and configure resources.

How much?

- There is no additional charge for using AWS CloudFormation, you pay for AWS resources created by it as if you had created them manually.

Reference:

FAQs

Category:

Compute

Amazon Elastic Beanstalk

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- AWS Elastic Beanstalk is an application management platform.
- It is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

Why?

- You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring.
- Elastic Beanstalk provides a unified user interface (UI) to monitor and manage the health of your applications.

When?

- Elastic Beanstalk is ideal if you have a PHP, Java, Python, Ruby, Node.js, .NET, Go, or Docker web application.
- You want to retain full control over the AWS resources powering your application and can access the underlying resources at any time.

Where?

- Amazon Elastic Beanstalk is a regional service.
- You can save your environment's configuration as an object in Amazon S3 bucket in a folder named after your application. It can be applied to other environments during environment creation, or applied to a running environment.

Who?

- You select the AWS resources, such as Amazon EC2 instance type including Spot instances, that are optimal for your application.
- You can choose to automatically get the latest platform and new patches using managed platform updates.

How?

- Elastic Beanstalk uses core AWS services such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), AWS Auto Scaling, and Elastic Load Balancing (ELB) to easily support applications that need to scale to serve millions of users.

How much?

- There is no additional charge for AWS Elastic Beanstalk. You pay for AWS resources (e.g. EC2 instances or S3 buckets) you create to store and run your application.
- You only pay for what you use, as you use it; there are no minimum fees and no upfront commitments.