

Deployment & Managing Infrastructure at Scale

There are several different ways in which you can deploy your workload and infrastructure to AWS.

CloudFormation

- IaC
- CloudFormation gives you a declarative way to outline your AWS infrastructure.
- For example, within a template you can declare:

I want a security group

I want 3 EC2 instances

I want 2 S3 Buckets

I want an ELB in front of all these machines

CloudFormation will create all this infrastructure for you, in exactly this order, with the exact configuration specified.

- CloudFormation supports only YAML and JSON for code if you use the console, but, the **AWS CDK** allows you to define CloudFormation templates using familiar programming languages such as Python, TypeScript, Java, C#, and others.

AWS Cloud Development Kit (CDK)



- Define your cloud infrastructure using a familiar language:
 - JavaScript/TypeScript, Python, Java, and .NET
- The code is "compiled" into a CloudFormation template (JSON/YAML)
- You can therefore deploy infrastructure and application runtime code together
 - Great for Lambda functions
 - Great for Docker containers in ECS / EKS



ChatGPT

"Application runtime code" refers to the portion of a software application that is responsible for executing and managing the actual logic and behavior of the application during runtime. In the context of software development, an application typically consists of two main components: the source code and the runtime environment.

Here's a breakdown of these components:

1. **Source Code:** This is the human-readable code that developers write to define the functionality of an application. It includes the algorithms, data structures, business logic, and other instructions that, when executed, produce the desired behavior. Source code is written in programming languages such as Python, Java, C++, JavaScript, etc.
2. **Runtime Code:** Once the source code is written, it needs to be executed to perform the intended tasks. The runtime code is the machine-readable version of the source code that is executed by the computer's hardware and software environment during the runtime of the application. This execution is where the application's behavior is manifested.

The runtime code is generated through a process called compilation or interpretation, depending on the programming language and the nature of the application. Compiled languages, like C++ or Java, are translated into machine code or bytecode before execution. Interpreted languages, like Python or JavaScript, are translated on-the-fly by an interpreter during runtime.

- Benefits

- **laC - Infrastructure as code** - no resources need to be provisioned manually, this allows more control. any changes to the configuration needs to be done through code.
- **Cost effective**
 - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
 - You can estimate the costs of your resources using the CloudFormation template
 - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely
- **Declarative programming** - you don't need to figure out what should be configured first (the order of things) as cloudFormation is smart enough to do it for you.
- Ability to destroy and recreate any infrastructure on the fly.
- Supports almost all AWS resources and you can use custom resources for the ones which are not supported.
- With CloudFormation, you can leverage the existing templates on the web or the documentation. You don't necessarily need to create one.

CloudFormation Stack Designer

- Example: WordPress CloudFormation Stack
- We can see all the resources
- We can see the relations between the components



CloudFormation Hands-on

CloudFormation > Stacks > Create stack

Step 1
Create stack

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

☒ Template is ready ☐ Use a sample template ☐ Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL
Provide an Amazon S3 URL to your template.

☒ Upload a template file
Upload your template directly to the console.

☐ Sync from Git - new
Sync a template from your Git repository.

Upload a template file
Choose file

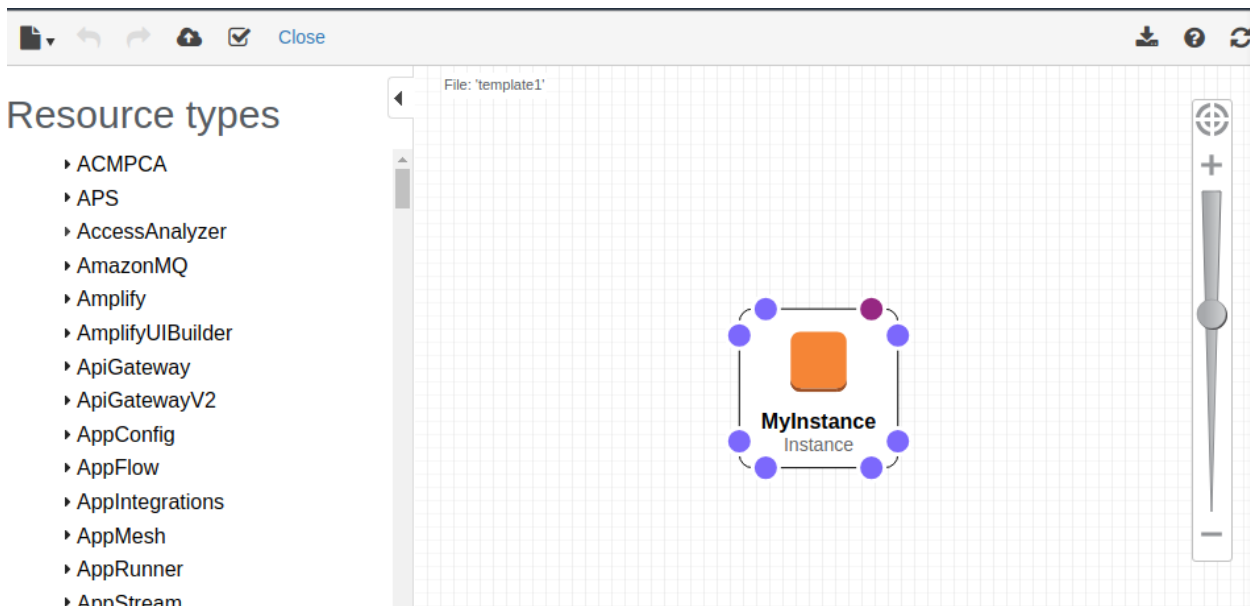
0-just-ec2.yaml
JSON or YAML formatted file

S3 URL: <https://s3.us-east-1.amazonaws.com/cf-templates-1twv5erg@nvgp-us-east-1/2023-11-30T175856.893Zfq-0-just-ec2.yaml>

[View in Designer](#)

Cancel **Next**

- You can click on “view in designer” to see the infrastructure created so far.



- Name the instance, give it a tag, leave the remaining settings as is.
- Once the stack is created, check the template, it consists of code. This is why CF is called IaC.

DemoCloudFormation



Delete

Update

Stack actions ▼

Create stack ▼

Stack info

Events

Resources

Outputs

Parameters

Template

Changesets

Git sync - new

Template

View in Designer

Copy to clipboard



```
---
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      AvailabilityZone: us-east-1a
      ImageId: ami-a4c7edb2
      InstanceType: t2.micro
```

=

DemoCloudFormation



Stack info

Events

Resources

Outputs

Parameters

Template

Changes



Events (5)

Detect root cause

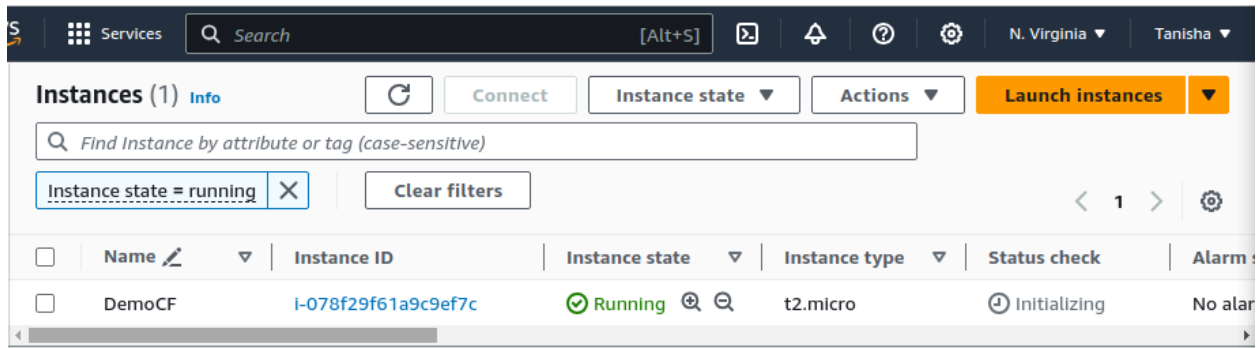


Search events



Timestamp ▼	Logical ID	Status	Status reason
2023-11-30 23:34:34 UTC+0530	DemoCloudFormation	✓ CREATE_COMPLETE	-
2023-11-30 23:34:33 UTC+0530	MyInstance	✓ CREATE_COMPLETE	-
2023-11-30 23:34:01 UTC+0530	MyInstance	ⓘ CREATE_IN_PROGRES S	Resource creation Initiated
2023-11-30 23:33:59 UTC+0530	MyInstance	ⓘ CREATE_IN_PROGRES S	-
2023-11-30 23:33:55 UTC+0530	DemoCloudFormation	ⓘ CREATE_IN_PROGRES S	User Initiated

- You can verify this by checking the EC2 dashboard.



- Lets update our stack.

Update stack

Prerequisite – Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

☐ Use current template
 ☒ Replace current template
 ☐ Edit template in Designer

Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL
 ☒ Upload a template file

Upload a template file

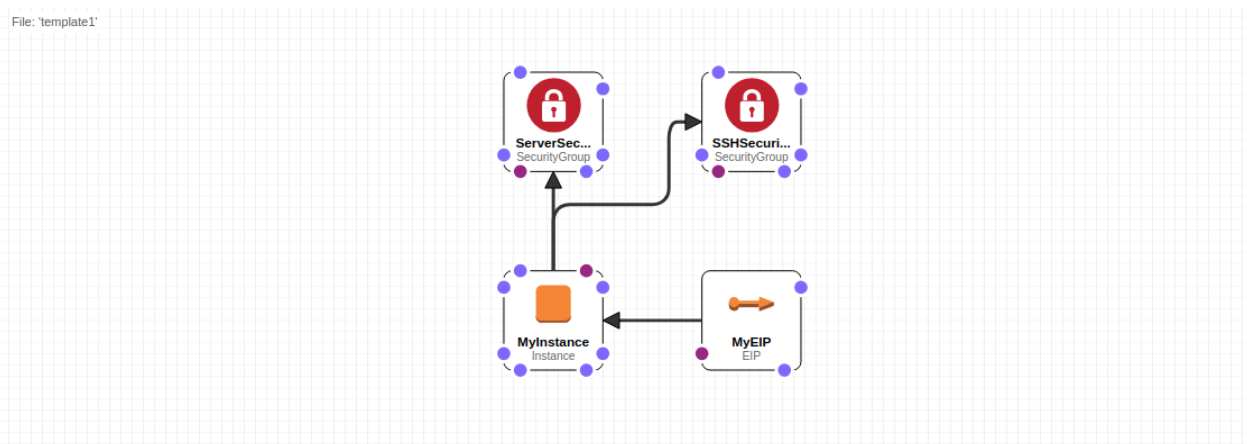
Choose file

1-ec2-with-sg-elb.yaml

JSON or YAML formatted file

S3 URL: <https://s3.us-east-1.amazonaws.com/cf-templates-1twv5erg9nvdp-us-east-1/2023-11-30T180940.622Zg3w-1-ec2-with-sg-elb.yaml>

[View in Designer](#)

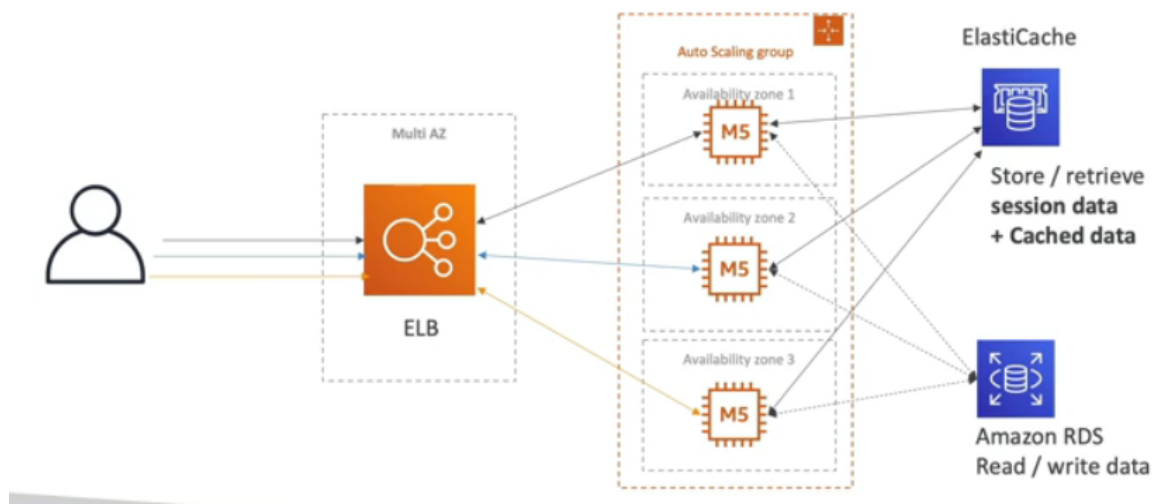


- can't proceed. error aa raha hai kuch.

Elastic Beanstalk

- When we deploy a web application to AWS, we typically follow a 3-tier architecture.

Typical architecture: Web App 3-tier



- As a developer on AWS, you don't want to worry about anything but the code.
- **Elastic Beanstalk** is a developer centric view of deploying an application on AWS.
- It uses all the components typically used for an application (ec2 instances, asg, elb, rds, etc.) but it's all in one view which makes it easy to understand for the developer.
- The developer can still have full control over the configuration.
- **It is a PaaS (Platform as a Service)**
- Beanstalk is free to use but you pay for the underlying instances.

- Managed service
 - Instance configuration / OS is handled by Beanstalk
 - Deployment strategy is configurable but performed by Elastic Beanstalk
 - Capacity provisioning
 - Load balancing & auto-scaling
 - Application health-monitoring & responsiveness
- Just the application code is the responsibility of the developer

Deployment architecture models offered by Beanstalk:

1. Single Instance Deployment

In the Single Instance deployment model, your application runs on a single Amazon EC2 instance. This setup is suitable for development or testing environments where high availability and scalability are not the primary concerns.

2. Load Balanced Auto Scaling

This model is designed for production workloads where higher availability and scalability are essential. It involves deploying the application across multiple EC2 instances behind an Elastic Load Balancer (ELB).

3. Auto Scaling Only

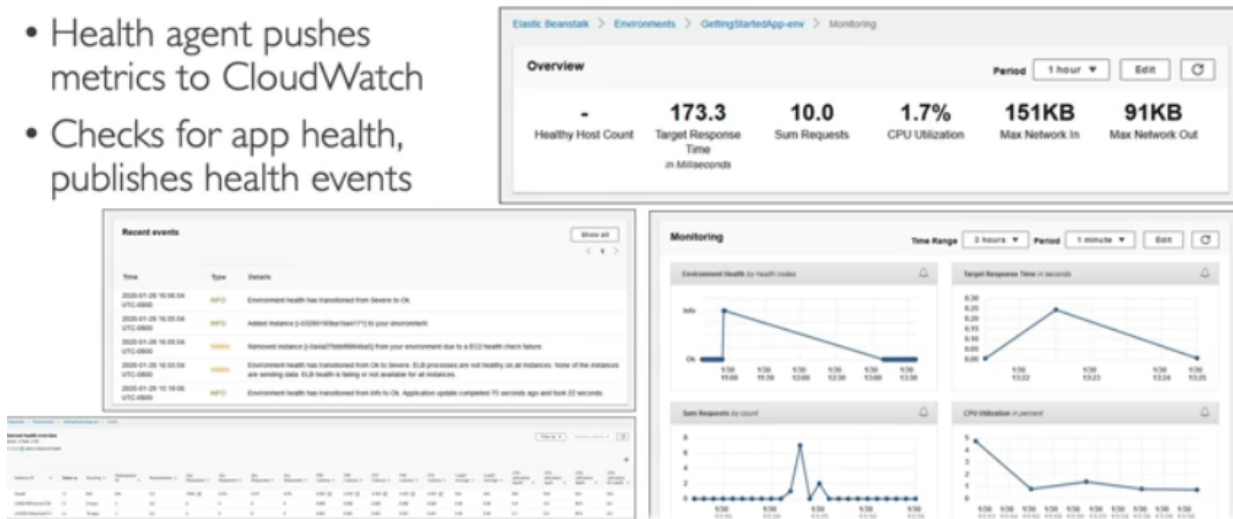
In this model, the application is deployed on multiple EC2 instances, but without an Elastic Load Balancer. It's suitable for scenarios where you want to scale horizontally based on demand but don't require load balancing.

For example, worker tasks or backend computation based tasks where there is no user-interaction, so you don't need a load balancer to direct any user requests. Another scenario could be when you want to scale horizontally (with ASGs) and are trying to cut some costs.

Health Monitoring in Beanstalk

- Beanstalk has a full monitoring service available within the service itself. Any instance that falls under Beanstalk will have a health agent which will push the performance metrics to CloudWatch. It will also check for app health.
- Beanstalk is a great way for you do health monitor your applications.

- Health agent pushes metrics to CloudWatch
- Checks for app health, publishes health events



Beanstalk Hands-on

Step 1

Configure environment

Step 2

Configure service access

Step 3 - optional

Set up networking, database,
and tags

Step 4 - optional

Configure instance traffic and
scaling

Step 5 - optional

Configure updates, monitoring,
and logging

Step 6

Review

Configure environment [Info](#)

Environment tier [Info](#)

Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

☒ **Web server environment**

Run a website, web application, or web API that serves HTTP requests. [Learn more](#) [↗](#)

☐ **Worker environment**

Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#) [↗](#)

Application information [Info](#)

Application name

Maximum length of 100 characters.

► **Application tags (optional)**

Platform [Info](#)

Platform type

☒ **Managed platform**

Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#) 

☐ **Custom platform**

Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

Node.js



Platform branch

Node.js 18 running on 64bit Amazon Linux 2023



Platform version

6.0.3 (Recommended)



Application code [Info](#)

☒ **Sample application**

☐ **Existing version**

Application versions that you have uploaded.

☐ **Upload your code**

Upload a source bundle from your computer or copy one from Amazon S3.

Presets Info

Start from a preset that matches your use case or choose custom configuration to unset recommended values and use the service's default values.

Configuration presets

☒ Single instance (free tier eligible)
 ☐ Single instance (using spot instance)
 ☐ High availability
 ☐ High availability (using spot and on-demand instances)
 ☐ Custom configuration

Cancel
Next

- Before proceeding, go to IAM in a new tab. Create a role for an EC2 instance. Allow the following permissions to the role.

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AdministratorAccess-AWSElasticBeanstalk	AWS managed	Grants account administrative permissions. Explicitly allows developers and administrators to gain direct access to re...
<input type="checkbox"/>	AWSElasticBeanstalkCustomPlatformforEC2Role	AWS managed	Provide the instance in your custom platform builder environment permission to launch EC2 instance, create EBS sna...
<input type="checkbox"/>	AWSElasticBeanstalkEnhancedHealth	AWS managed	AWS Elastic Beanstalk Service policy for Health Monitoring system
<input type="checkbox"/>	AWSElasticBeanstalkManagedUpdatesCustomerRolePo...	AWS managed	This policy is for the AWS Elastic Beanstalk service role used to perform managed updates of Elastic Beanstalk enviro...
<input checked="" type="checkbox"/>	AWSElasticBeanstalkMulticontainerDocker	AWS managed	Provide the instances in your multicontainer Docker environment access to use the Amazon EC2 Container Service to ...
<input type="checkbox"/>	AWSElasticBeanstalkReadOnly	AWS managed	Grants read-only permissions. Explicitly allows operators to gain direct access to retrieve information about resources...
<input type="checkbox"/>	AWSElasticBeanstalkRoleCore	AWS managed	AWSElasticBeanstalkRoleCore (Elastic Beanstalk operations role) Allows core operation of a web service environment.
<input type="checkbox"/>	AWSElasticBeanstalkRoleCWL	AWS managed	(Elastic Beanstalk operations role) Allows an environment to manage Amazon CloudWatch Logs log groups.
<input type="checkbox"/>	AWSElasticBeanstalkRoleECS	AWS managed	(Elastic Beanstalk operations role) Allows a multicontainer Docker environment to manage Amazon ECS clusters.
<input type="checkbox"/>	AWSElasticBeanstalkRoleRDS	AWS managed	(Elastic Beanstalk operations role) Allows an environment to integrate an Amazon RDS instance.
<input type="checkbox"/>	AWSElasticBeanstalkRoleSNS	AWS managed	(Elastic Beanstalk operations role) Allows an environment to enable Amazon SNS topic integration.
<input type="checkbox"/>	AWSElasticBeanstalkRoleWorkerTier	AWS managed	(Elastic Beanstalk operations role) Allows a worker environment tier to create an Amazon DynamoDB table and an A...
<input checked="" type="checkbox"/>	AWSElasticBeanstalkWebTier	AWS managed	Provide the instances in your web server environment access to upload log files to Amazon S3.
<input checked="" type="checkbox"/>	AWSElasticBeanstalkWorkerTier	AWS managed	Provide the instances in your worker environment access to upload log files to Amazon S3, to use Amazon SQS to mo...

- Now, go back to Beanstalk creation dashboard. Select the created IAM role for "EC2 instance profile"

Step 2

Configure service access

Step 3 - optional

[Set up networking, database, and tags](#)

Step 4 - optional

[Configure instance traffic and scaling](#)

Step 5 - optional

[Configure updates, monitoring, and logging](#)

Step 6

[Review](#)

Service access

IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

Service role

☒ Create and use new service role

☐ Use an existing service role

Service role name

Enter the name for an IAM role that Elastic Beanstalk will create to assume as a service role. Beanstalk will attach the required managed policies to it.

[View permission details](#)

EC2 key pair

Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

EC2 instance profile

Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

[View permission details](#)













Cancel

[Skip to review](#)

[Previous](#)

[Next](#)

- Leave everything else as it is and create the beanstalk environment.
- You can monitor the events of your environment. These events are basically whatever is occurring in the stack created in CloudFormation which is allowing us to provision these resources behind the scenes.

Events (9) Info			
<input type="text" value="Filter events by text, property or value"/>			 1  
Time	Type	Details	
December 1, 2023 16:43:41 (UTC+5:30)	 INFO	Successfully launched environment: MyWebApp-dev	
December 1, 2023 16:43:40 (UTC+5:30)	 INFO	Application available at MyWebApp-dev.eba-mpwwwmaun.ap-south-1.elasticbeanstalk.com.	
December 1, 2023 16:43:26 (UTC+5:30)	 INFO	Instance deployment completed successfully.	
December 1, 2023 16:42:53 (UTC+5:30)	 INFO	Waiting for EC2 instances to launch. This may take a few minutes.	
December 1, 2023 16:42:22 (UTC+5:30)	 INFO	Created EIP: 3.111.216.43	
December 1, 2023 16:42:12 (UTC+5:30)	 INFO	Environment health has transitioned to Pending. Initialization in progress (running for 17 seconds). There are no instances.	
December 1, 2023 16:42:07 (UTC+5:30)	 INFO	Created security group named: awseb-e-qdh2sp8cfs-stack-AWSEBSecurityGroup-113T5ROGAO3FE	
December 1, 2023 16:41:42 (UTC+5:30)	 INFO	Using elasticbeanstalk-ap-south-1-734605198697 as Amazon S3 storage bucket for environment	

- You can now access the website deployed using the DNS provided by beanstalk.

MyWebApp-dev [Info](#)



Actions ▼

Upload and deploy

Environment overview

Health

✓ Ok

Environment ID

📄 e-qdh2sp8cfs

Domain

[MyWebApp-dev.eba-mpwwmaun.ap-south-1.elasticbeanstalk.com](#) [🔗](#)

Application name

[MyWebApp](#)

Platform

Change version

Platform

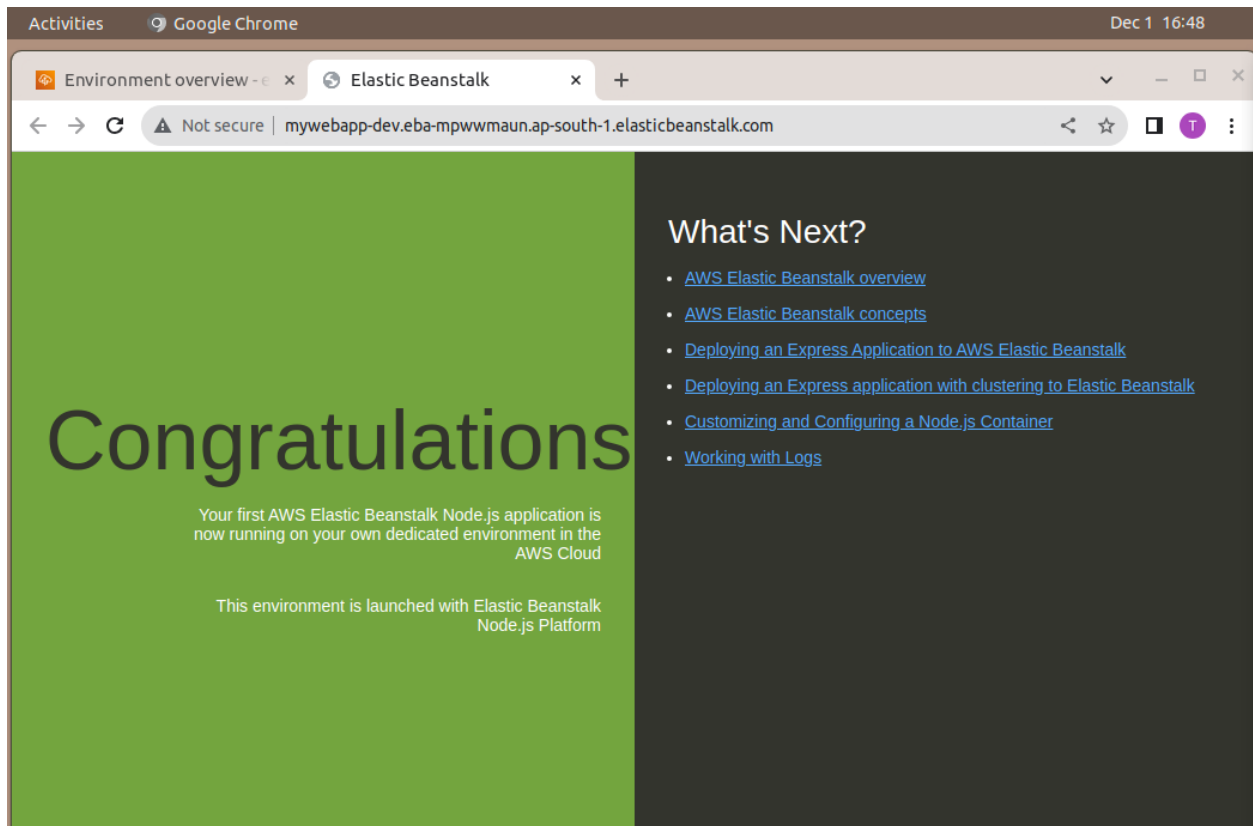
Node.js 18 running on 64bit Amazon Linux 2023/6.0.3

Running version

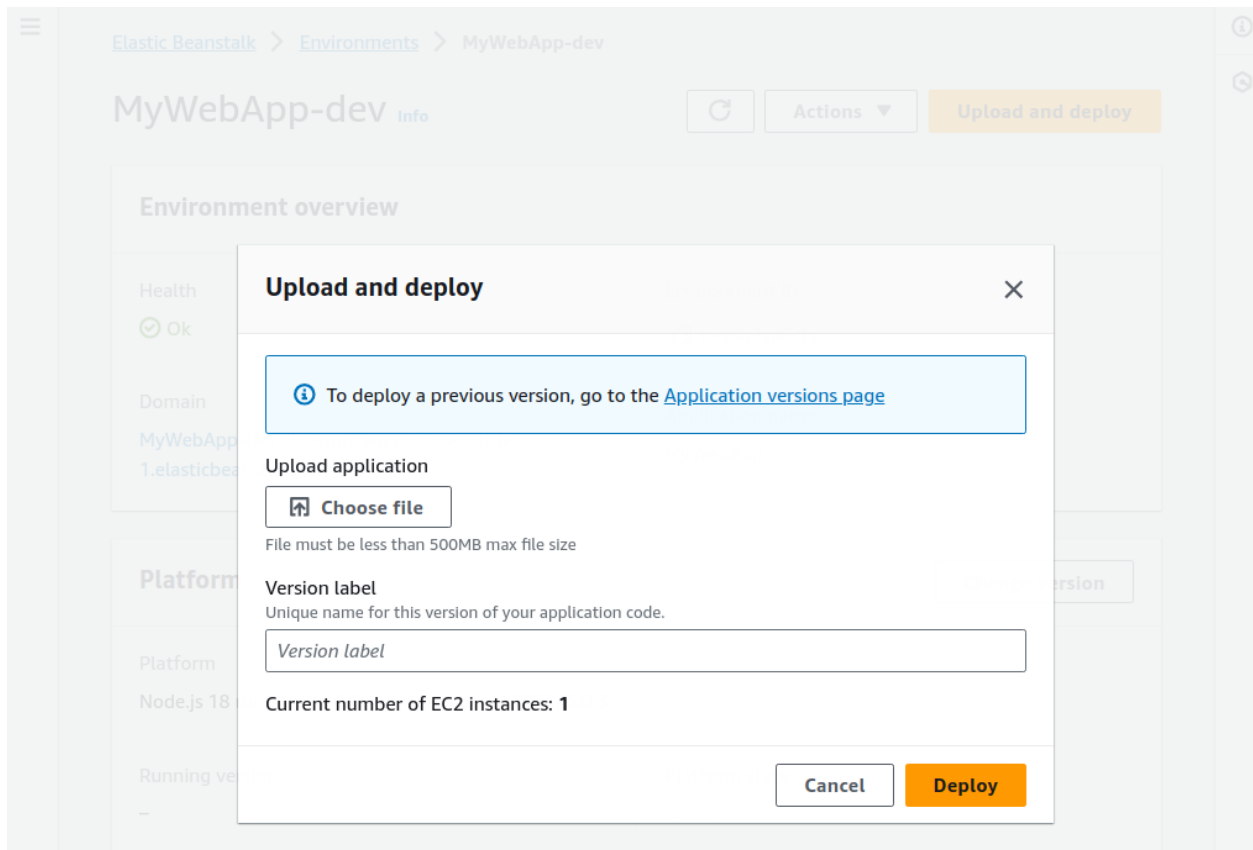
–

Platform state

✓ Supported



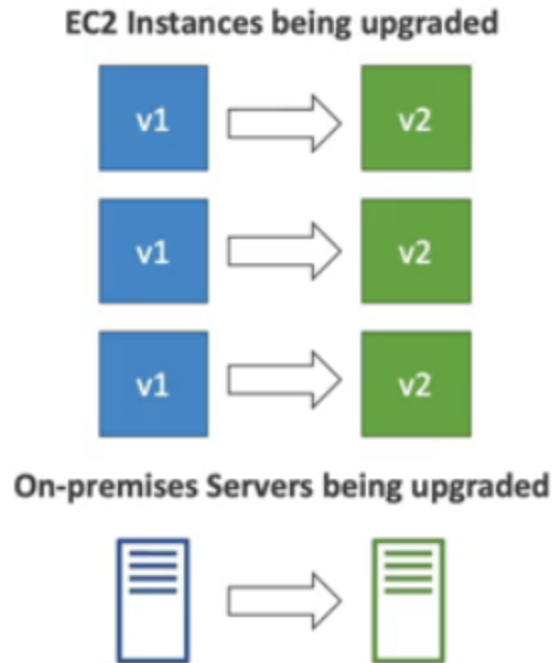
- You also have the options to deploy new or previous versions of your application.



AWS Code Tools - Developer Tools

AWS CodeDeploy

- A fully managed deployment service provided by (AWS) that automates the process of deploying applications to a variety of compute services such as Amazon EC2 instances, AWS Lambda functions, and on-premises servers.
- It is a hybrid service as it works with both on-premises servers and ec2 instances.



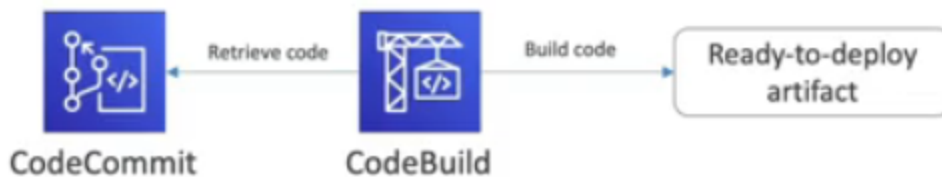
- One thing to note is that, servers or instances must be provisioned and configured with CodeDeploy Agent ahead of time.

AWS CodeCommit

- Before pushing the code to the servers, it needs to be stored somewhere.
- Developers want to store the code in a git-based repository (for easy collaboration globally) and usually opt for GitHub for the same but AWS has a competing product called CodeCommit.
- CodeCommit offers a source-controlled service which can be used to host git-based repositories.
- Any code changes are automatically versioned.
- Advantage over github is, CodeCommit is fully managed, private, secure and integrated with AWS.

AWS CodeBuild

- Allows you to build your code in the cloud.
- It compiles your source code, run tests, and produces packages that are ready to be deployed.
- So, as a developer you need to worry only about coding and making sure that a service within AWS build your code every time you push code updates into your CodeCommit repository.



- You were confused if CodeBuild lets you create or write code all in the cloud.



ChatGPT

AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready for deployment. However, CodeBuild is primarily focused on building and testing your code rather than being a code editor.



You

Replying to:

CodeBuild is primarily focused on building and testing your code rather than being a code editor

what exactly do you mean by "building" my code?



ChatGPT

In the context of software development, "building" refers to the process of transforming your human-readable source code into executable code or other deployable artifacts. The build process involves several steps, including compilation (for languages like Java or C++), linking, running tests, and packaging the application for deployment. The purpose of building your code is to ensure that it's in a format that can be executed or deployed to various environments.

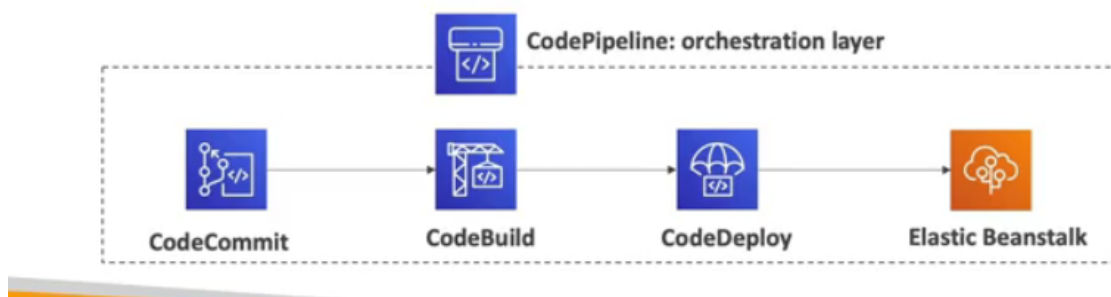
3. Code Editing:

- While CodeBuild doesn't provide a code editor itself, it is designed to work with your source code stored in a repository.
- You typically write and edit your code using your preferred local development environment (e.g., IDE on your machine).

- The purpose of using a service like AWS CodeBuild is to automate build steps in a cloud environment. CodeBuild takes your build specifications, which you define in a `buildspec.yml` file, and executes the necessary build and test processes in a managed environment.
- This automation helps ensure consistency in your build process, allows for parallel processing, and integrates seamlessly into CI/CD pipelines, where changes to your code trigger automated builds and deployments.

AWS CodePipeline

- CodePipeline is basically a way for you to orchestrate the different steps involved in automatically pushing your code to production.
- For example, code → build → test → provision required servers → deploy. This could be a codepipeline defined by you.



- This concept is the basis for CI/CD (continuous integration & continuous delivery)
- Orchestration of codepipeline = aws CodePipeline service

- Benefits:
 - Fully managed, compatible with CodeCommit, CodeBuild, CodeDeploy, Elastic Beanstalk, CloudFormation, GitHub, 3rd-party services (GitHub...) & custom plugins...
 - Fast delivery & rapid updates

AWS CodeArtifact

- Software packages often have dependencies. To store and retrieve these dependencies is called artifact management.
- Traditionally, you need to set up your own artifact management system which could get complex as it involves various packages depending on each other.
- So, AWS offers CodeArtifact which is a secure and scalable artifact management system for software development.
- It works with common dependencies tools like npm, yarn, pip, etc.
- Developers and CodeBuild can then retrieve dependencies straight from CodeArtifact.

AWS CodeStar

- CodeStar is basically a unified user-interface to easily manage software development activities all at one place.
- CodeStar is your one shot way to **quick start** your development project. Once you start with it, CodeStar will automatically give you a neat dashboard which will be representing whatever is going behind the scenes AND it will set up a CodeCommit repository, a CodeBuild build process, a CodeDeploy, a CodePipeline, even any instances which need to be provisioned.

AWS Cloud9

- Cloud9 is a cloud IDE used for writing, running and debugging code in the cloud.

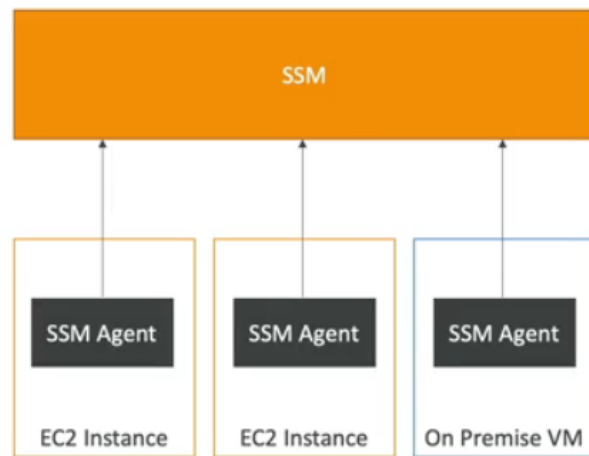
- A cloud IDE can be used within a web browser.
- This means you can work on your projects from your office, home or anywhere just with internet access and without the need to set up.
- Allows code collaboration in real-time (pair programming).

AWS SSM (Systems Manager)

- SSM helps you manage a fleet of your ec2 instances and on-premises systems at scale. This makes it a hybrid service.
- It's basically a systems manager which gives you insights about the state of your infrastructure.
- Most imp features:
 - Automatic patching of all the servers and instances for enhanced compliance.
 - Run commands across an entire fleet of servers
 - Store parameter configurations with SSM Parameter Store
- Keywords from exam perspective:
 - patching of fleet of ec2 instances or on-premises servers
 - or running a command consistently across all servers

How Systems Manager works

- We need to install the SSM agent onto the systems we control
- Installed by default on Amazon Linux AMI & some Ubuntu AMI
- If an instance can't be controlled with SSM, it's probably an issue with the SSM agent!
- Thanks to the SSM agent, we can run commands, patch & configure our servers



SSM Session Manager

- With SSM Session manager, you can start a secure shell on your ec2 instance and on-premises servers without having to SSH into them.
- Therefore, no SSH access or keys needed.
- No port 22 (better security) needed which implies better security.
- Sends session log data into S3 or CloudWatch Logs.
- Fleet Manager is a service where all the ec2 instances associated with SSM will be shown. They are called managed nodes.
- Hands on given in the lecture

SSM Parameter Store

- The SSM Parameter Store is a way for you to store configuration and secrets securely on AWS.
- You can store anything you want like API Keys, passwords, etc.
- It is super secure as you can control access to each parameter using IAM

- Also, your configurations have version tracking and option for encryption.

AWS OpsWorks

- OpsWorks was created by AWS to provide managed Chef & Puppet in the cloud.
 - Chef & Puppet help you perform server configuration automatically, or repetitive actions
 - They work great with EC2 & On-Premises VM
- It is an alternative to AWS SSM.
- It can only provision standard AWS resources like EC2, Databases, Load Balancers, etc.
- For exam, keywords are: **Chef & Puppet = OpsWorks** and a situation where you were already using Chef & Puppet before migrating to cloud and now you want to reuse those templates in the cloud.



Summary

Deployment - Summary

- **CloudFormation:** (AWS only)
 - Infrastructure as Code, works with almost all of AWS resources
 - Repeat across Regions & Accounts
- **Beanstalk:** (AWS only)
 - Platform as a Service (PaaS), limited to certain programming languages or Docker
 - Deploy code consistently with a known architecture: ex, ALB + EC2 + RDS
- **CodeDeploy** (hybrid): deploy & upgrade any application onto servers
- **Systems Manager** (hybrid): patch, configure and run commands at scale
- **OpsWorks** (hybrid): managed Chef and Puppet in AWS

Developer Services - Summary

- **CodeCommit:** Store code in private git repository (version controlled)
- **CodeBuild:** Build & test code in AWS
- **CodeDeploy:** Deploy code onto servers
- **CodePipeline:** Orchestration of pipeline (from code to build to deploy)
- **CodeArtifact:** Store software packages / dependencies on AWS
- **CodeStar:** Unified view for allowing developers to do CI/CD and code
- **Cloud9:** Cloud IDE (Integrated Development Environment) with collab
- **AWS CDK:** Define your cloud infrastructure using a programming language



Good job!

The AWS Cloud Development Kit (AWS CDK) is an open source software development framework to define your cloud application resources using familiar programming languages.

Question 6:

A developer would like to deploy infrastructure on AWS but only knows Python. Which AWS service can assist him?



Software Development Kit (SDK)



Cloud Development Kit (CDK)



CloudFormation



CodeBuild



Good job!

Elastic Beanstalk is a Platform as a Service (PaaS). You only manage data and applications. AWS Elastic Beanstalk makes it even easier for developers to quickly deploy and manage applications in the AWS Cloud

Question 9:

Which service is referred to as a Platform as a Service (PaaS)?



Elastic Beanstalk



OpsWorks



CloudFormation



EC2

**Good job!**

AWS CloudFormation templates are JSON or YAML-formatted text files. They are declarations of the AWS resources that make up a stack.

Question 10:

What is called the declaration of the AWS resources that make up a stack?



CloudFormation Schemas



CloudFormation Diagrams



CloudFormation Templates



CloudFormation Models

**Good job!**

AWS CodeArtifact is a fully managed artifact repository (also called code dependencies) service that makes it easy for organizations of any size to securely store, publish, and share software packages used in their software development process.

Question 11:

Which of the following services can a developer use to store code dependencies?



CodeBuild



CodeCommit



Cloud9



CodeArtifact

**Good job!**

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodeStar is used to quickly develop, build, and deploy applications on AWS with a unified user interface.

Question 12:

CodeStar can orchestrate the different steps to have code automatically pushed to production, while CodePipeline is a unified UI to easily manage software development activities in one place.

☐ True

☒ False

**Good job!**

CloudFormation and Elastic Beanstalk are free of use, but you do pay for the resources created.

Question 14:

CloudFormation and Elastic Beanstalk are free of use.

☒ True

☐ False

