Choose a Topic

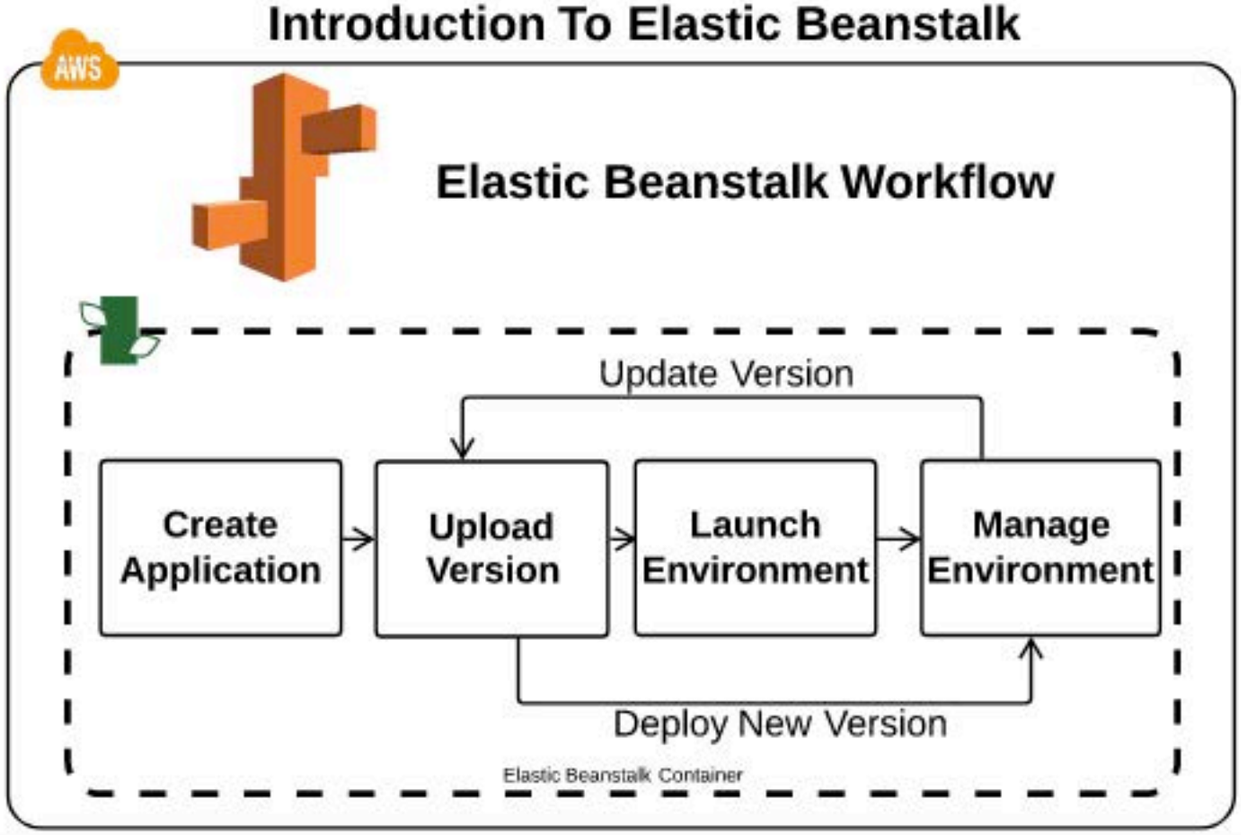
- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Application Deployments On Elastic Beanstalk

Scenario: With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring. This section will go into the details necessary for the DevOps Professional to successfully utilize Elastic Beanstalk.

- Introduction
- Deployment Strategies
- Docker With Elastic Beanstalk
- Elastic Beanstalk With RDS
- Environment Configurations
- Using With CloudFormation
- Elastic Beanstalk Scenarios

Introduction To Elastic Beanstalk



Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Application Deployments On Elastic Beanstalk

X

AWS

Elastic Beanstalk Key Concepts

- **Makes it easy to quickly deploy applications on AWS without having to worry about infrastructure**: Create the app --> Deploy the App --> EB provisions capacity, sets up load balancing and auto scaling, and configures monitoring --> You manage the environment and deploy new apps.
- Elastic Beanstalk Use Cases:
  - You want to spend minimal time learning/configuring infrastructure.
  - Quick prototyping and testing.
  - Shorter application lifecycles.
  - Maintain some flexibility over the resources powering your apps.
- Elastic Beanstalk anti-patterns:
  - When you need complete control over resource configurations.
  - Existing apps can be difficult to fit in the Elastic Beanstalk model.
  - Can get complicated if you have a lot of dependencies.
- Elastic Beanstalk Components:
  - Application - the collection of components such as environments, versions, and configurations.
  - Application Version - Part of an App. Apps can have multiple versions. You can deploy multiple versions to test them.
- Environment - Version that is deployed with AWS resources.
- Environment Configuration - Setting and parameters that define the environment and resources.
- Configuration Template - Used to create repeatable environments configurations.

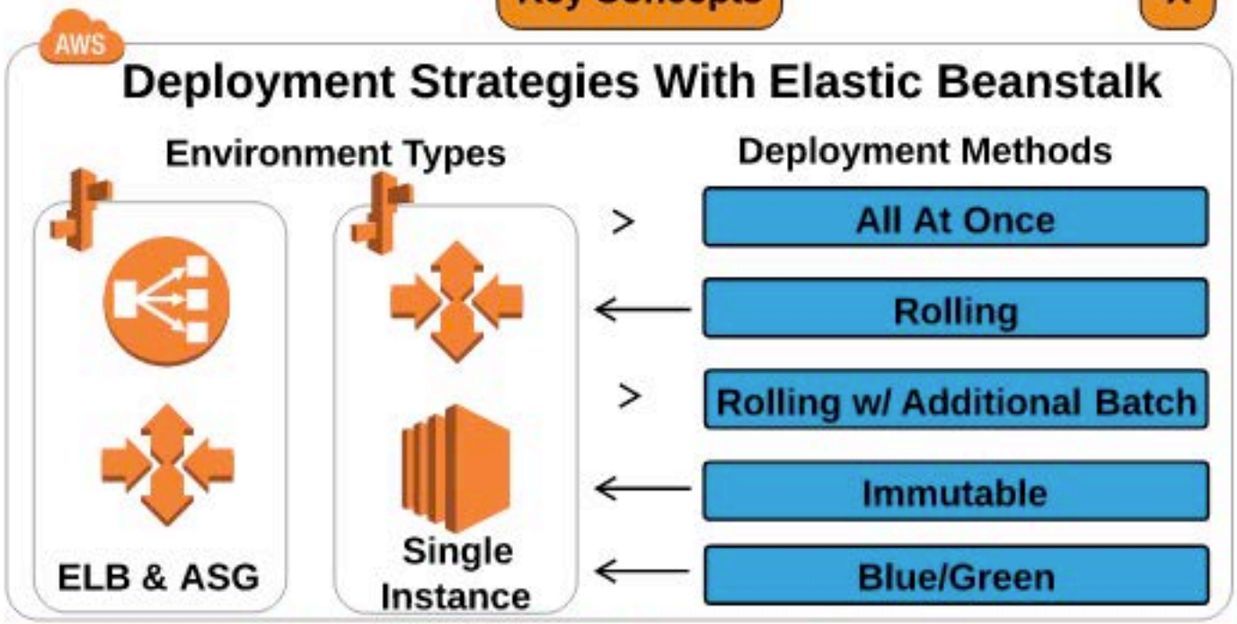
Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Application Deployments On Elastic Beanstalk

Key Concepts

X



Elastic Beanstalk Deployment Strategies

- Environment Types:
  - Load Balancing and Auto Scaling
    - Auto Scaling handles launching, configuring, and terminating instances.
    - The ELB handles client requests and communicates with back-end instances.
  - Single Instance Evironments
    - One instance with an Elastic IP Address.
    - The Auto Scaling Group keeps one instance running.
    - No Load Balancer
- **Deployment Methods**: All at once, Rolling, Rolling With Additional Batch, Immutable, Blue/Green
- **All At Once** - updates all existing instances at the same time with an in-place update.
  - Pros - Fastest Method, Requires no DNS changes
  - Cons - can cause downtime
  - On deployment failure - redeploy a working version
- **Rolling Deployment:**
  - Pros: prevents downtime, control over number of instances update, uses health checks, requires no DNS changes.
  - Cons: If deployment fails halfway through instances serve different versions of the app. Can cause capacity issues.
  - On deployment failure: redeploy with another Rolling Deployment.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Application Deployments On Elastic Beanstalk

X

AWS

Elastic Beanstalk Deployments - continued

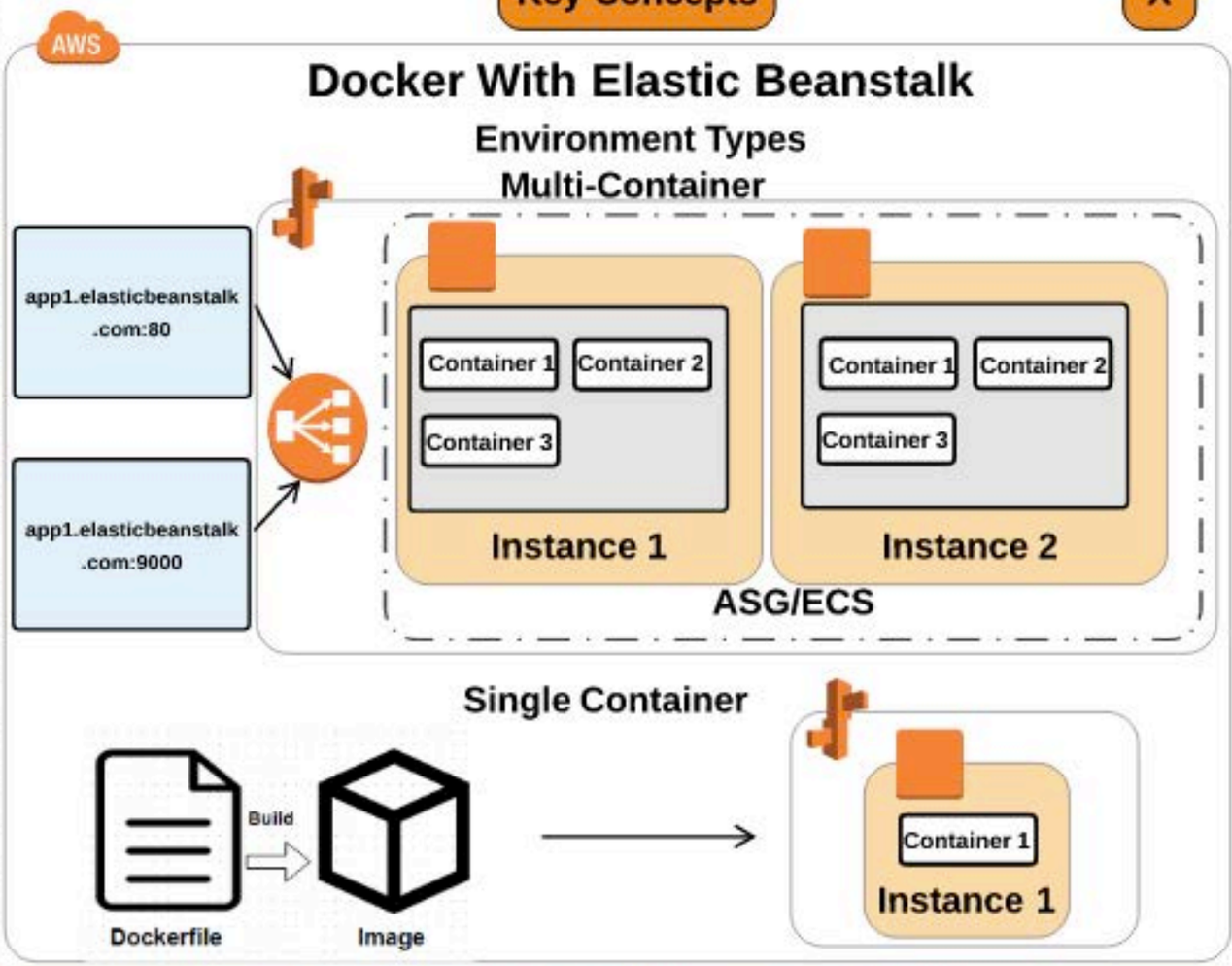
- **Rolling with additional batch deployment** - updates one batch of instances at a time, starting with a batch of new instances not in rotation.
  - Pros - Same benefits as rolling deployment. Prevents capacity issues.
  - Cons - can take longer than the rolling deployment.
  - On deployment failure - Re-deploy with another rolling with additional batch deployment.
- **Immutable Deployment** - Replace existing instances with new instances by creating a temporary ASG, testing one instance with the new app, then adding the new instances to the original ASG while terminating the original instances and the temporary ASG.
  - Pros - Prevents downtime, uses new resources instead of in-place updates, simple and clean rollback.
  - Cons - Doubles the number of instances for a short period.
  - On deployment failure - Terminate the temporary ASG, and redeploy.
- **Blue/Green Deployment** - Replaces all resources including the ELB, ASG, and instances.
  - Pros - Prevents downtime, uses new resources instead of in-place updates, can test updates in an isolated environment.
  - Cons - requires a DNS name change. Doubles the number of instances while deployments are running.
  - On deployment failure - swap urls.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Key Concepts

X



Elastic Beanstalk Deployment Strategies

- Docker is an open-source project that provides a layer of abstraction and automation of operating system-level virtualization on Linux.
- Docker containers provide a tool to package an application and all of its dependencies in a virtual container so that it can be run on any Linux system.
- We can configure, manage, package, and deploy Docker images. These images give us consistency.
- Scenarios:
  - Multiple developers working on the same app but on different machines and possibly different OSs. Create a Docker image of the app and send to all developers creating an identical evironment.
  - Migrating an existing application running on a custom application server to AWS. Package the app and depencies with Docker. Deploy with EB.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

X

AWS

Docker Deployments With EB - continued

- Docker can be used with Elastic Beanstalk to:
  - Maintain consistency across multiple uses.
  - Abstract the application from underlying virtualized hardware.
  - Answer the question: "Do I have all I need to run my application reliably?".
- There are 2 generic configurations for Docker with EB:
  - Single Container - Used to deploy a Docker image and source code inside a single container per instance.
  - Multicontainer - Used to deploy multiple containers per instance. Uses the Amazon ECS to deploy a cluster in the EB environment. Example: PHP in one container and Nginx in another.
- The single container environment can have both a Dockerfile and a Dockerrun.aws.json file. Dockerfile describes the image to build with instructions.
- Dockerrun.aws.json can specify the image and EB configurations.
- Multicontainer deployment uses the Dockerrun.aws.json file (required).
  - This deployment does not support custom images with Dockerfiles. Instead, we need to host images in public or private repositories:
    - Build the custom image
    - Deploy the image to an online repo
    - Create the Elastic Beanstalk environment
  - In addition to the Dockerrun file, we can also have configuration files in the .ebextensions directory. Example, we can create an ELB with a listener for port 8080:

```
option_settings :
  aws:elb:listener:8080:
  ListenerProtocol:  HTTP
  InstanceProtocol:  HTTP
  InstancePort:  8080
```

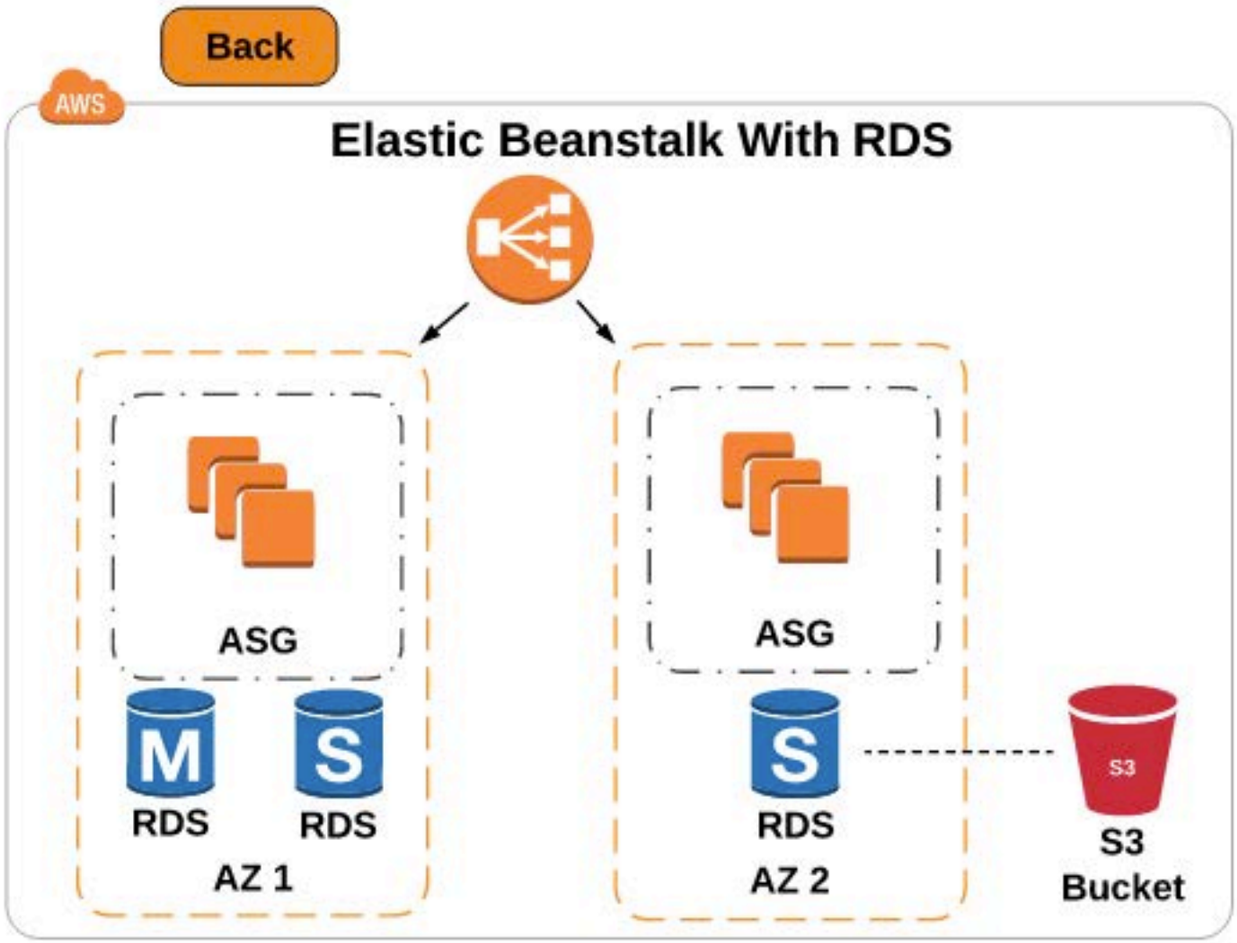
The map that port to our container from the host instance in the Dockerrun file:

```
"portMappings": [
  {  "hostPort": 8080,
     "containerPort": 8080
  }
  ]
```

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Back



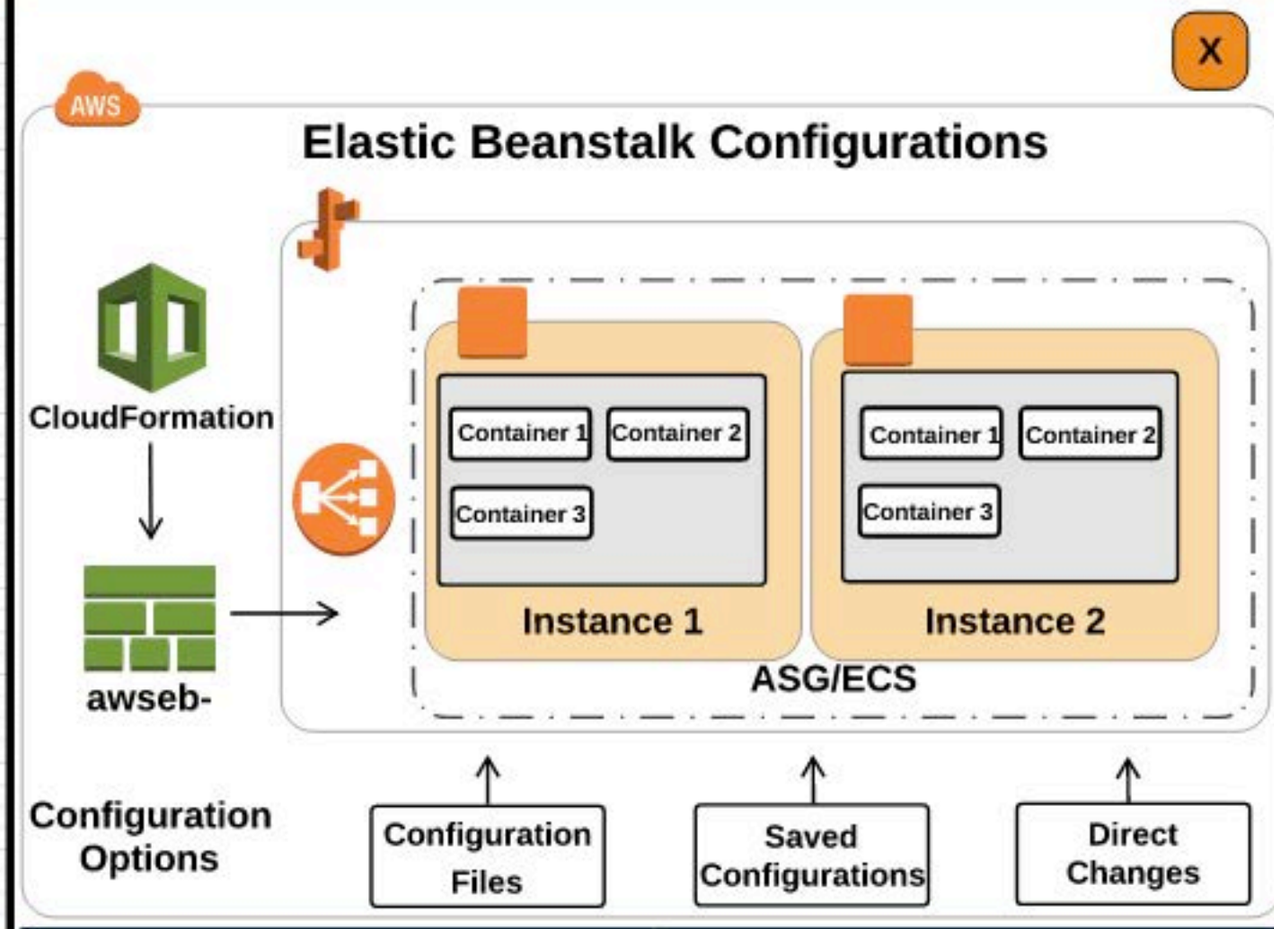
Elastic Beanstalk With RDS

- Elastic Beanstalk provides support for running RDS instances in your Elastic Beanstalk environment. This is great for dev and test environments.
- This is not ideal for a Prod environment because it ties the lifecycle of the DB to the lifecycle of the application.
- To decouple your database instance from your environment, you can run a database instance in Amazon RDS and configure your application to connect to it on launch.
- This enables you to connect multiple environments to a database, terminate an environment without affecting the database, and perform seamless updates with blue-green deployments.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Application Deployments On Elastic Beanstalk



Elastic Beanstalk Configurations

- Need a way to configure our environment to provide everything the app needs.
  - Configuration Files (in .ebextensions), Saved Configurations, Direct Changes
- Types of Configurations:
  - Deployment Options, Software Dependencies, CloudWatch Metrics and Alarms, Autoscaling and other service configs, Creating other resources (SQS, SNS, RDS).
- Anatomy of Configuration Files
  - **option_settings** - defines values for configuration settings. Can configure:
    - EB environment, AWS resources in our environment, software on instances.
  - **resources** - lets us define and customize resources in our environment.
  - **Other sections** - Commands, Container Commands, Files, Groups, Packages, Services, Sources, Users.
- Saved Configurations - Configurations can be saved and stored as object in S3. Can be used to save settings we've applied to an existing environment during or after environment configuration.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

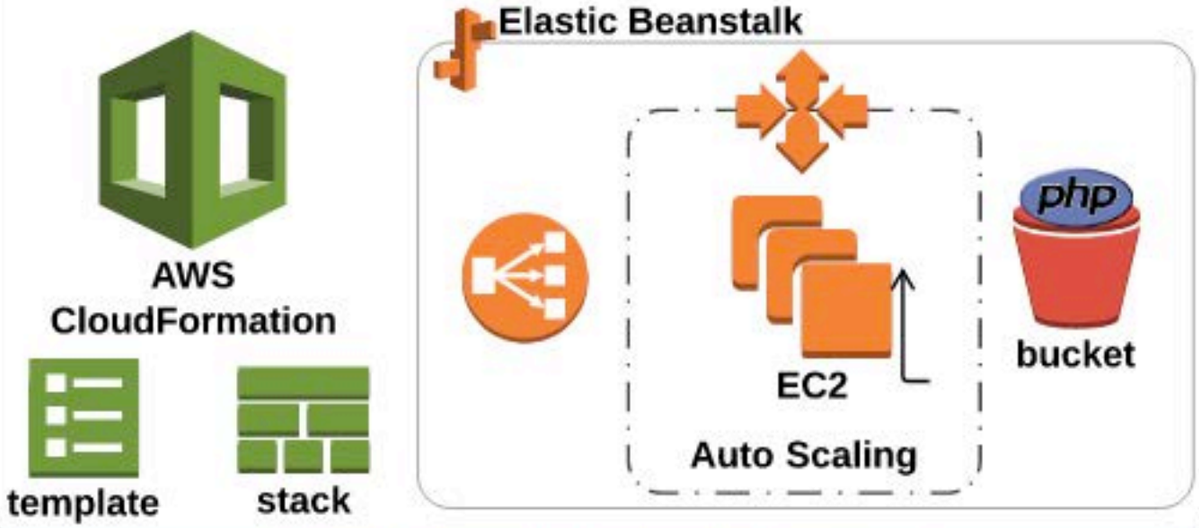
Application Deployments On Elastic Beanstalk

Key Concepts

X

AWS

Using Elastic Beanstalk with CloudFormation



Elastic Beanstalk With CloudFormation

- **CloudFormation -** You want a simple, yet finely controllable way to model and version collections of AWS resources.
- **Elastic Beanstalk** - You want to quickly get code running scalably and reliably without worrying about the underlying resources.
- Combine the features of Elastic Beanstalk with CloudFormation to make a great team.
- With CloudFormation, keep your infrastructure definition under source control and make it easy to evolve.
- Test changes to your infrastructure on staging accounts and review them before deployment.
- No undocumented changes can happen in your infrastructure.
- You can rollback to previous versions of you infrastructure.
- Elastic Beanstalk provides a great environment to run and operate your applications.
- Elastic Beanstalk allows you to test various versions of your application, and when paired with CloudFormation, you can quickly adjust your infrastructure to meet the needs of your app while maintaining robust configuration management.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Application Deployments On Elastic Beanstalk

X

AWS

Elastic Beanstalk Deployments - continued

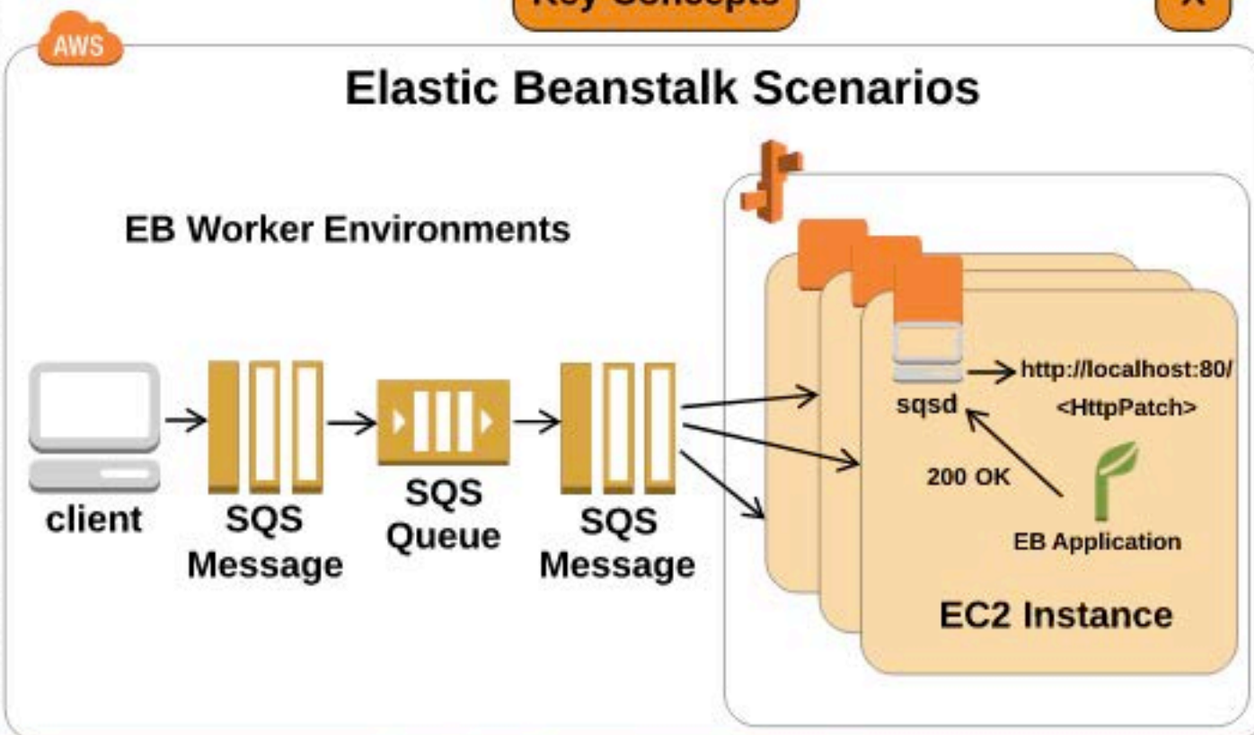
- **Rolling with additional batch deployment** - updates one batch of instances at a time, starting with a batch of new instances not in rotation.
  - Pros - Same benefits as rolling deployment. Prevents capacity issues.
  - Cons - can take longer than the rolling deployment.
  - On deployment failure - Re-deploy with another rolling with additional batch deployment.
- **Immutable Deployment** - Replace existing instances with new instances by creating a temporary ASG, testing one instance with the new app, then adding the new instances to the original ASG while terminating the original instances and the temporary ASG.
  - Pros - Prevents downtime, uses new resources instead of in-place updates, simple and clean rollback.
  - Cons - Doubles the number of instances for a short period.
  - On deployment failure - Terminate the temporary ASG, and redeploy.
- **Blue/Green Deployment** - Replaces all resources including the ELB, ASG, and instances.
  - Pros - Prevents downtime, uses new resources instead of in-place updates, can test updates in an isolated environment.
  - Cons - requires a DNS name change. Doubles the number of instances while deployments are running.
  - On deployment failure - swap urls.

Choose a Topic

- Introduction
- Auto Scaling Deployment Concepts
- Deployment Concepts With EC2
- CloudWatch For DevOps
- CloudFormation For DevOps
- Elastic Beanstalk For DevOps
- Application Deployments With OpsWorks
- DynamoDB Concepts
- S3 Concepts For DevOps
- Blue/Green Deployments
- Scenario Solver
- Deployment Pipelines
- API Gateway
- Lambda

Key Concepts

X



Elastic Beanstalk Scenarios

Elastic Beanstalk Worker Environments

- If your application performs operations or workflows that take a long time to complete, you can offload those tasks to a dedicated *worker environment*.
- One option is to spawn a worker process locally, return success, and process the task asynchronously.
- To avoid running long-running tasks locally, you can use the AWS SDK for your programming language to send them to an Amazon SQS queue, and run the process that performs them on a separate set of instances.
- The worker instances take items from the queue only when they have capacity to run them, preventing them from becoming overwhelmed.
- Elastic Beanstalk simplifies this process by managing the Amazon SQS queue and running a daemon process on each instance that reads from the queue for you.

Container Commands

- You can use the container_commands key to execute commands that affect your application source code. Container commands run after the application and web server have been set up and the application version archive has been extracted, but before the application version is deployed.
- You can use leader_only to only run the command on a single instance, or configure a test to only run the command when a test command evaluates to true.