



ondia



# Elastic Beanstalk



# AGENDA



- ▶ **Introduction to Elastic Beanstalk**
- ▶ **Basic concepts of Elastic Beanstalk**



# Introduction to Elastic Beanstalk

# Introduction to Elastic Beanstalk

## What is Elastic Beanstalk ?



- AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services.
- It is a kind of orchestration service offered by Amazon Web Services used to set up your application architecture.

# Introduction to Elastic Beanstalk



## What is Elastic Beanstalk ?

### ★ Favorites

Add favorites by clicking on the star next to the service name.

### Recently visited

- EC2
- CloudFormation
- Elastic Beanstalk
- Console Home
- API Gateway
- Lambda
- IAM
- S3
- Serverless Application Repository
- DynamoDB
- Route 53
- VPC
- Billing

### All services

#### Compute

- EC2
- Lightsail
- Lambda
- Batch
- Elastic Beanstalk
- Serverless Application Repository
- AWS Outposts
- EC2 Image Builder

#### Storage

- S3
- EFS
- FSx
- S3 Glacier
- Storage Gateway
- AWS Backup

#### Customer Enablement

- AWS IQ
- Support
- Managed Services
- Activate for Startups

#### Blockchain

- Amazon Managed Blockchain

#### Satellite

- Ground Station

#### Quantum Technologies

- Amazon Braket

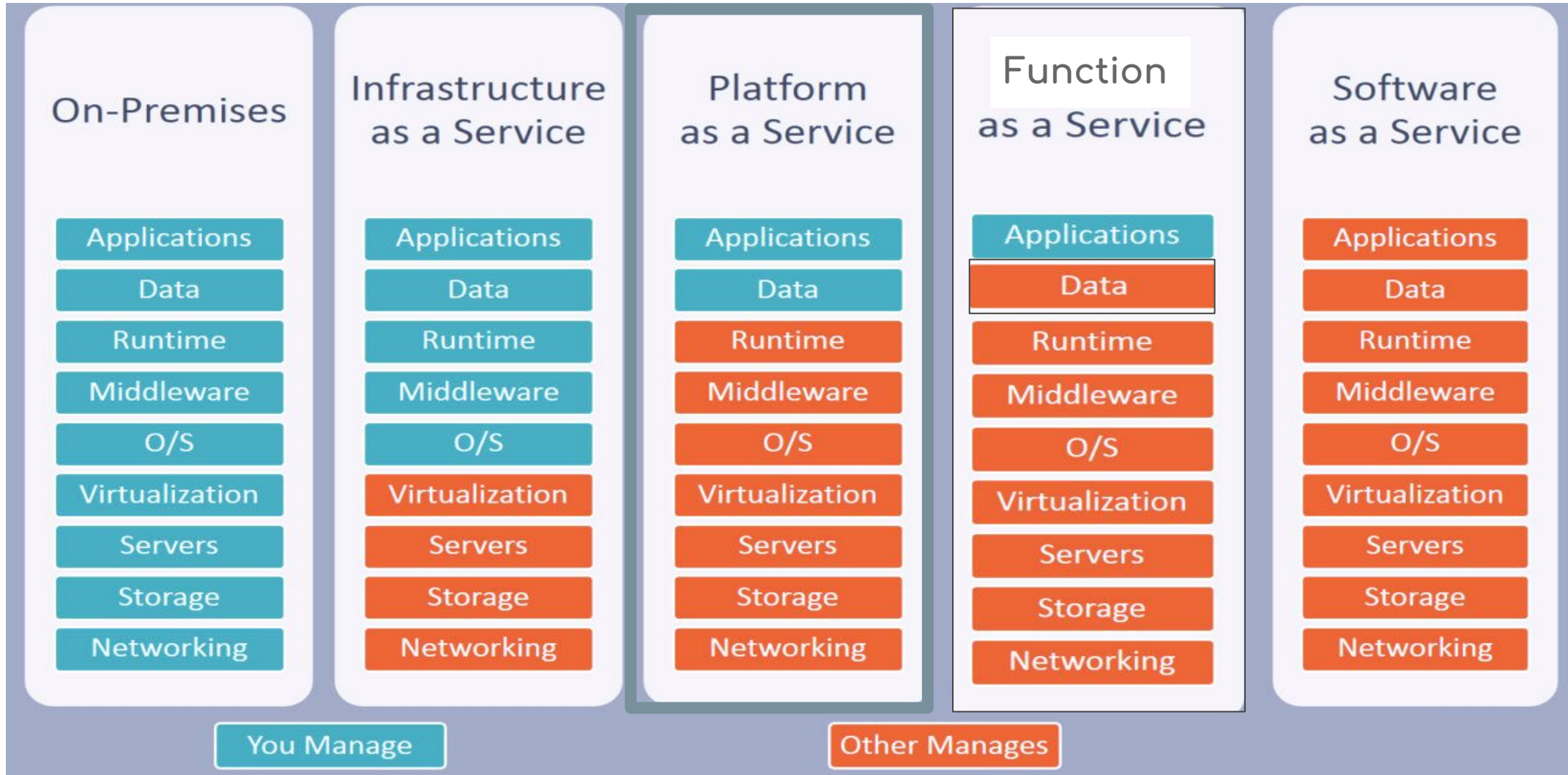
#### Management & Governance

- AWS Organizations

# Introduction to Elastic Beanstalk



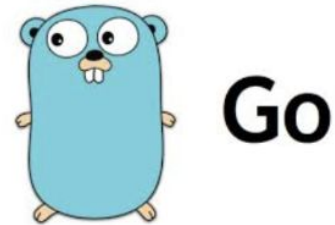
## What is Elastic Beanstalk ?



# Introduction to Elastic Beanstalk



## What is Elastic Beanstalk ?



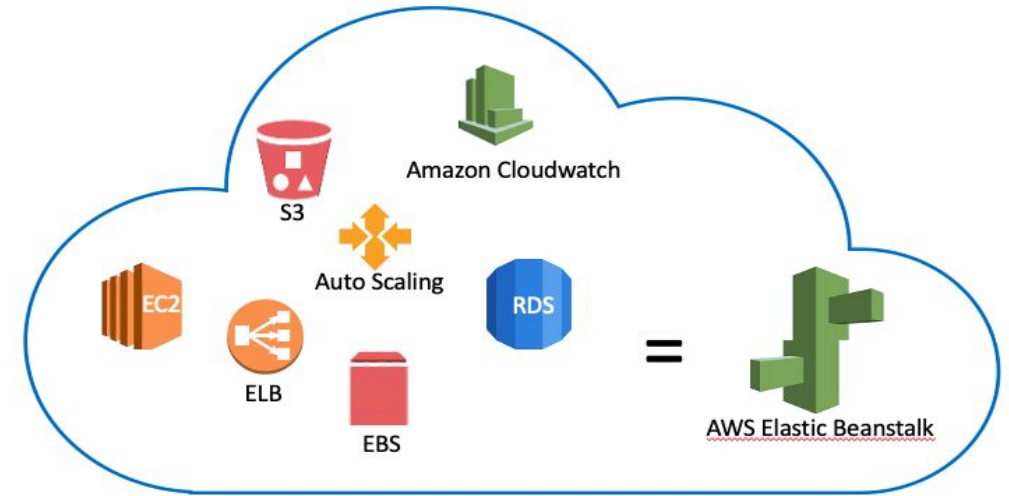


# Introduction to Elastic Beanstalk

## Why AWS Elastic Beanstalk?



- Automates the details of capacity provisioning,
- Load balancing,
- Auto scaling,
- Application deployment,



# Introduction to Elastic Beanstalk



- Automates management tasks:

- Monitoring,
- Version deployment,
- Health check
- Log



Monitoring



Health Check





# Basic Concepts of Elastic Beanstalk

# Basic Concepts of Elastic Beanstalk

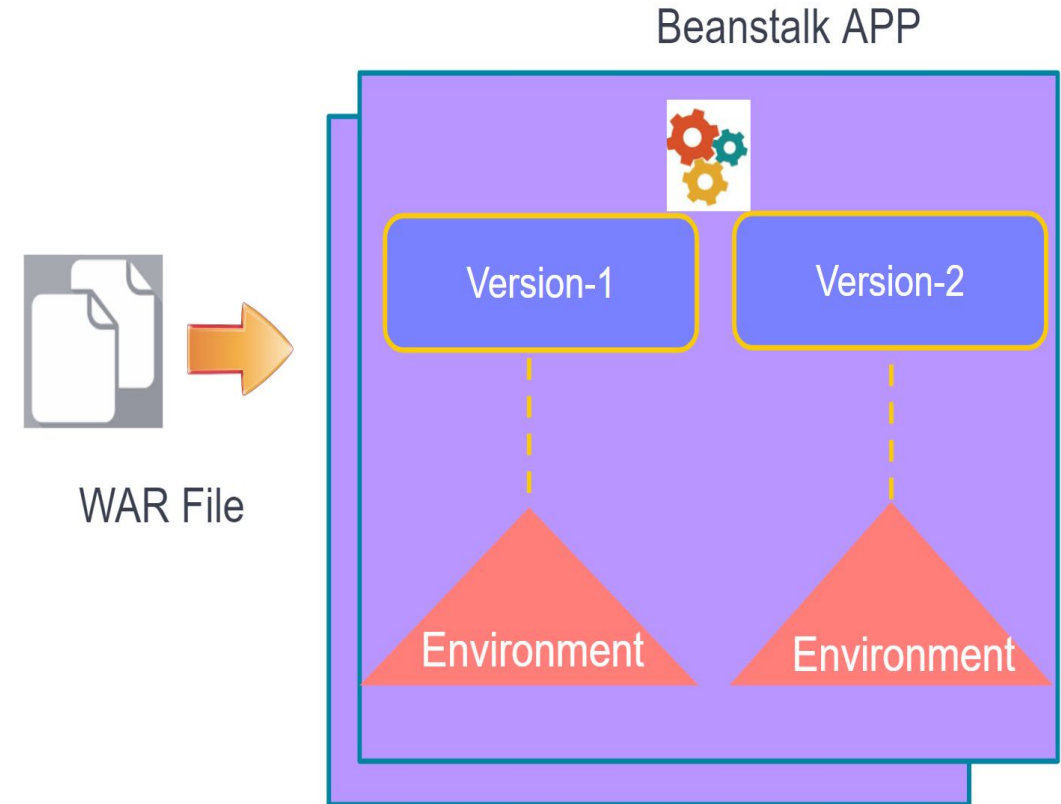


## Application

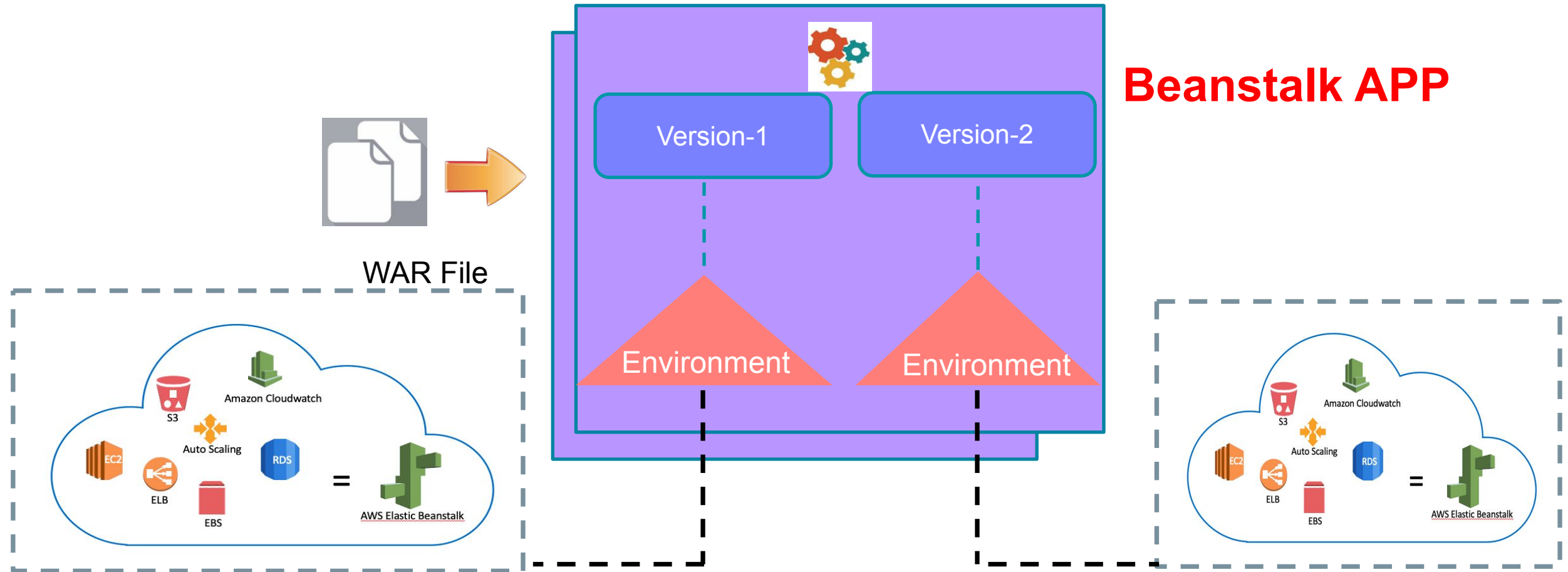
- **Application is a logical collection** of Elastic Beanstalk components. It covers all components.

## Application version

- Specific, labeled iteration of deployable code for a web application.



# Basic Concepts of Elastic Beanstalk



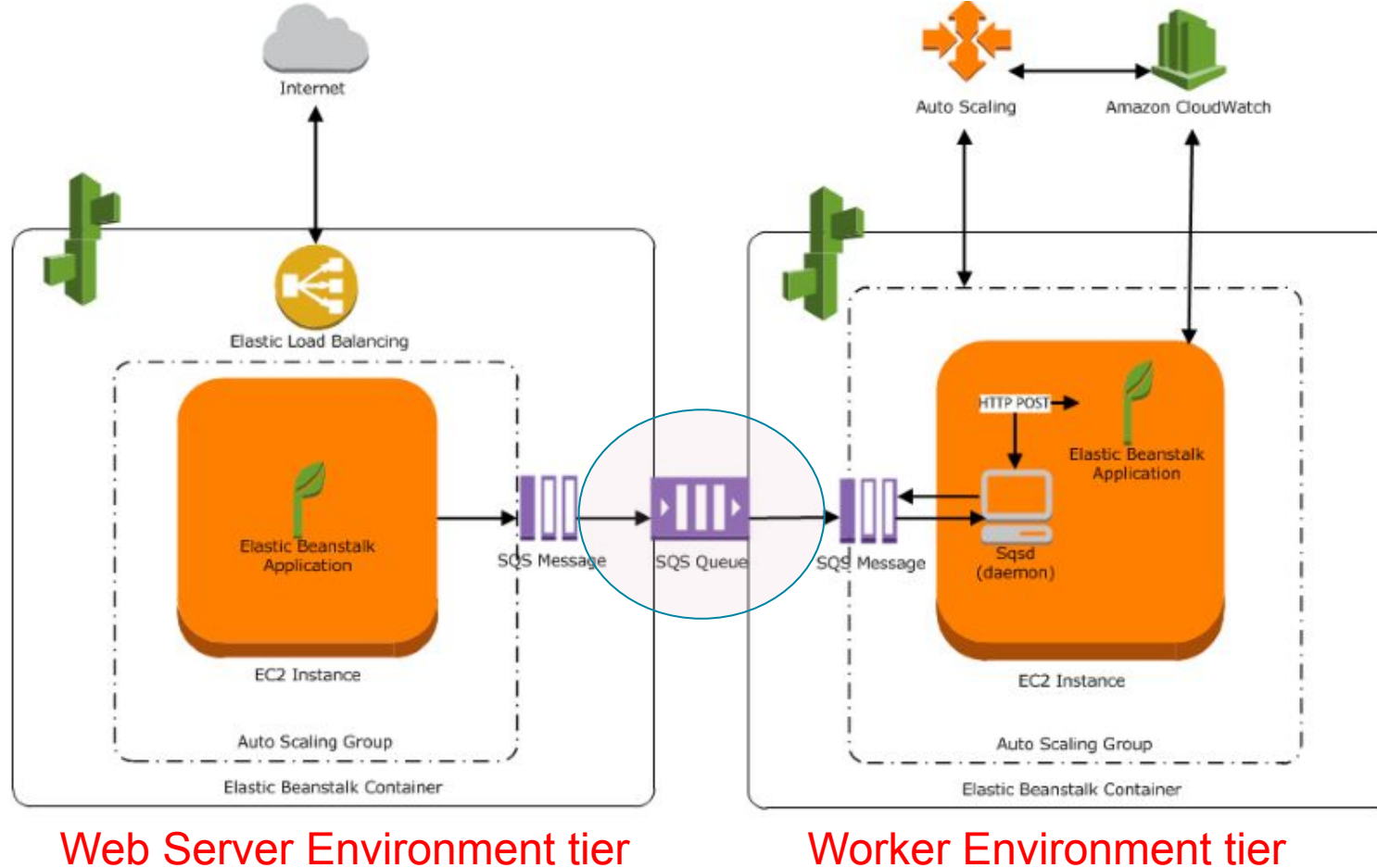
## Environment

- An environment is **a collection of AWS resources** running an application version. Each environment runs only one application version at a time.

# Basic Concepts of Elastic Beanstalk



## Environment Tier:



- The environment tier designates the type of application that the environment runs, and determines what resources Elastic Beanstalk provisions to support it.

# Basic Concepts of Elastic Beanstalk



Platform:

Platform
Platform
Java ▼
Platform branch
Corretto 11 running on 64bit Amazon Linux 2 ▼
Platform version
3.0.3 (Recommended) ▼

## Supported platform versions

- Docker
- Multicontainer Docker
- Preconfigured Docker
- Go
- Java SE
- Tomcat
- .NET Core on Linux
- .NET on Windows Server
- Node.js
- PHP
- Python
- Ruby

# Summary of Terms / Concepts



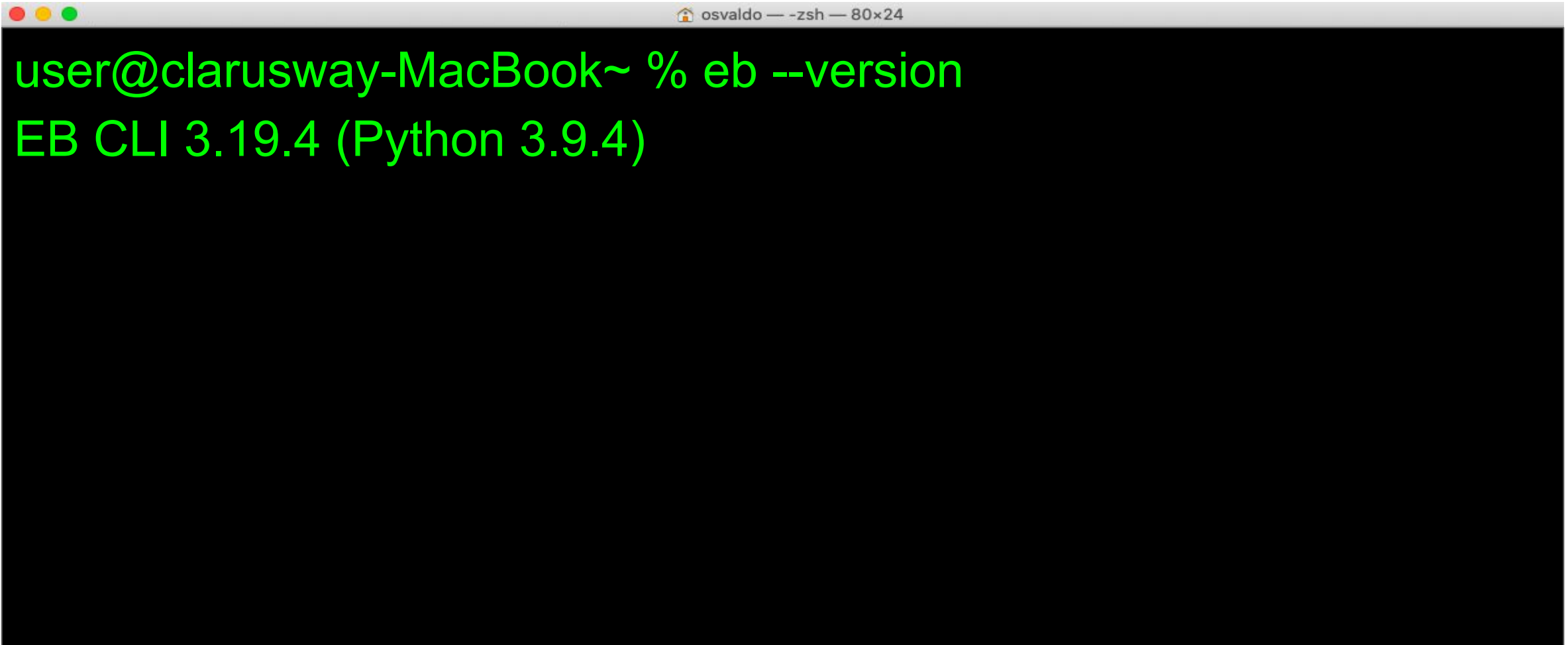
Concept	What it Means
Application	Logical collection of Elastic Beanstalk components required for a working deployment
Application Version	A labelled version of an application (e.g. 1.0, 1.1, 2.0, etc...)
Environment	A set of AWS resources running a specific application version (e.g. DEV, TEST, PROD)
Environment Tier	The type of application that an environment runs (either Web or Worker)
Platform	Combination of OS, programming language, web server - i.e. the “technology stack”



# Basic Concepts of Elastic Beanstalk

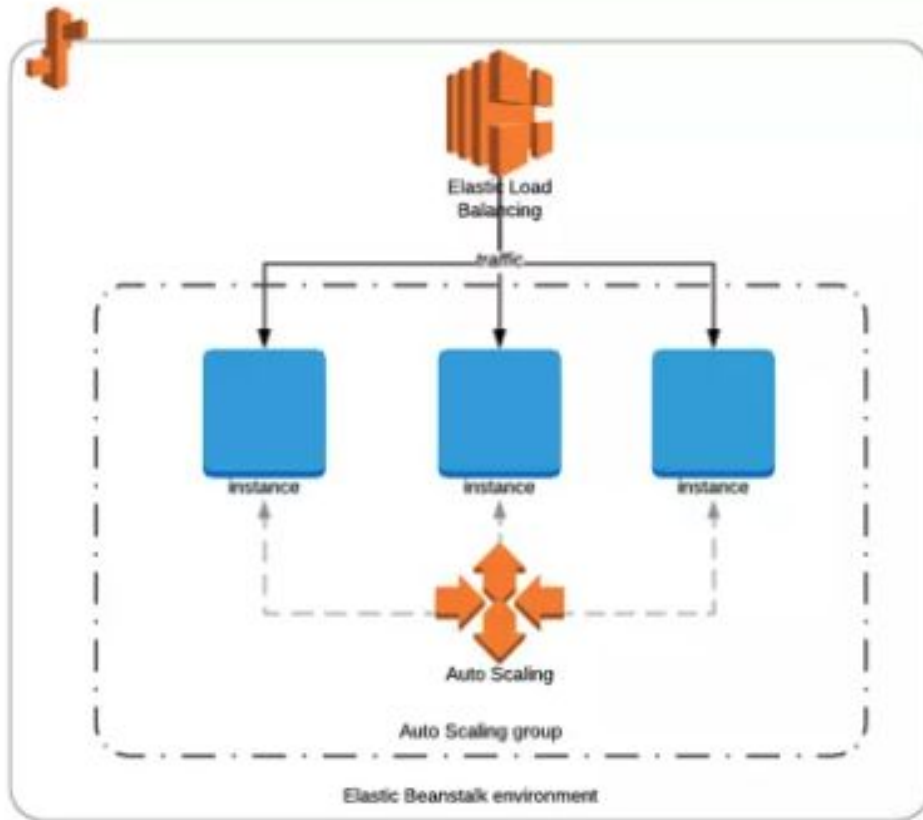


## Elastic Beanstalk Command Line Interface (EB CLI)

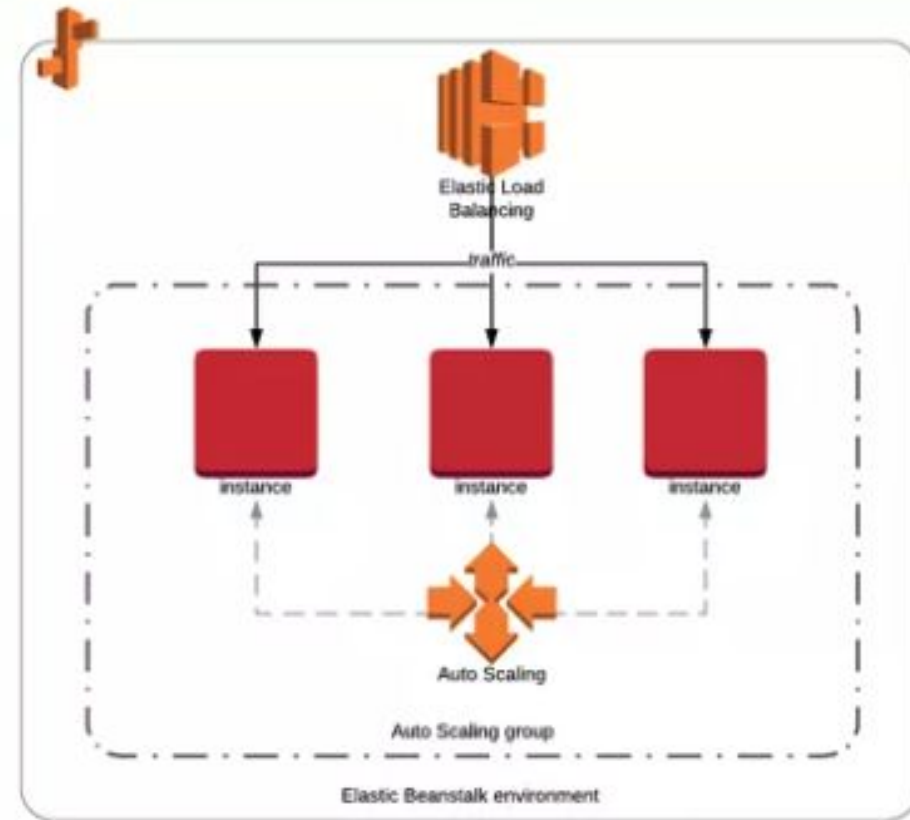
A terminal window with a dark background and light green text. The window title bar shows 'osvaldo — -zsh — 80x24'. The prompt is 'user@clarusway-MacBook~ %'. The command entered is 'eb --version'. The output is 'EB CLI 3.19.4 (Python 3.9.4)'.

```
osvaldo — -zsh — 80x24
user@clarusway-MacBook~ % eb --version
EB CLI 3.19.4 (Python 3.9.4)
```

# Deployment Models - All at once

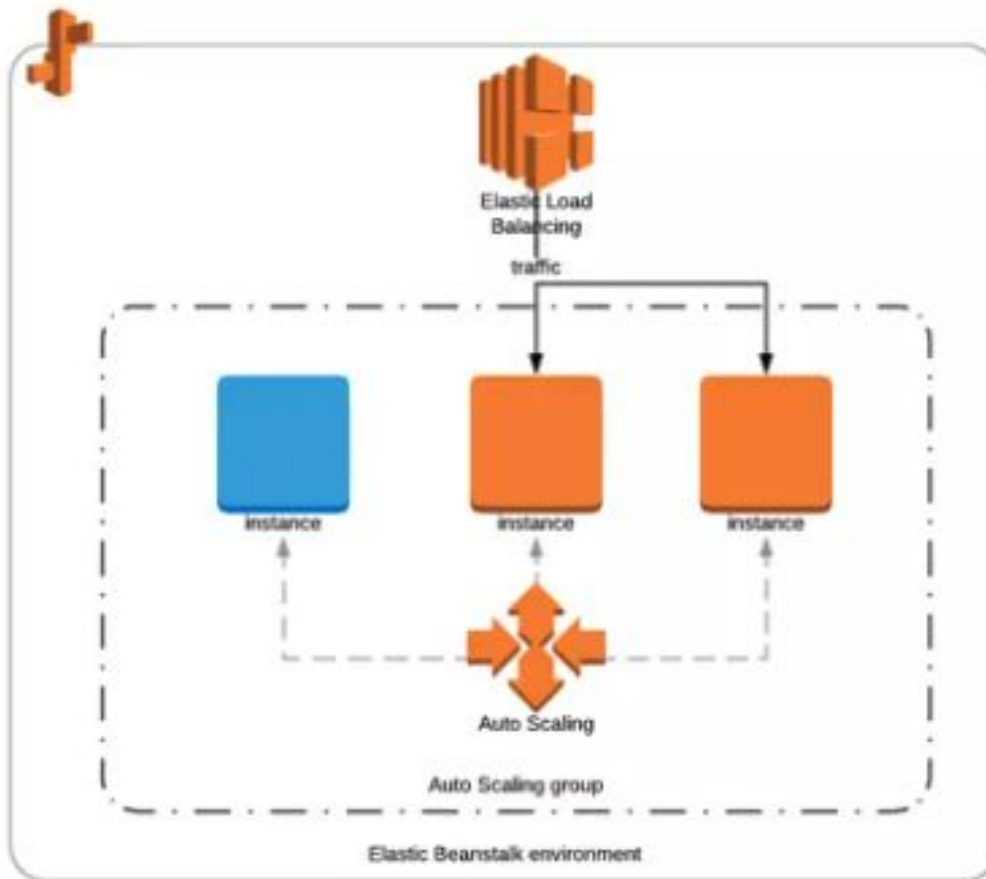


1. Deployment starts in all instances

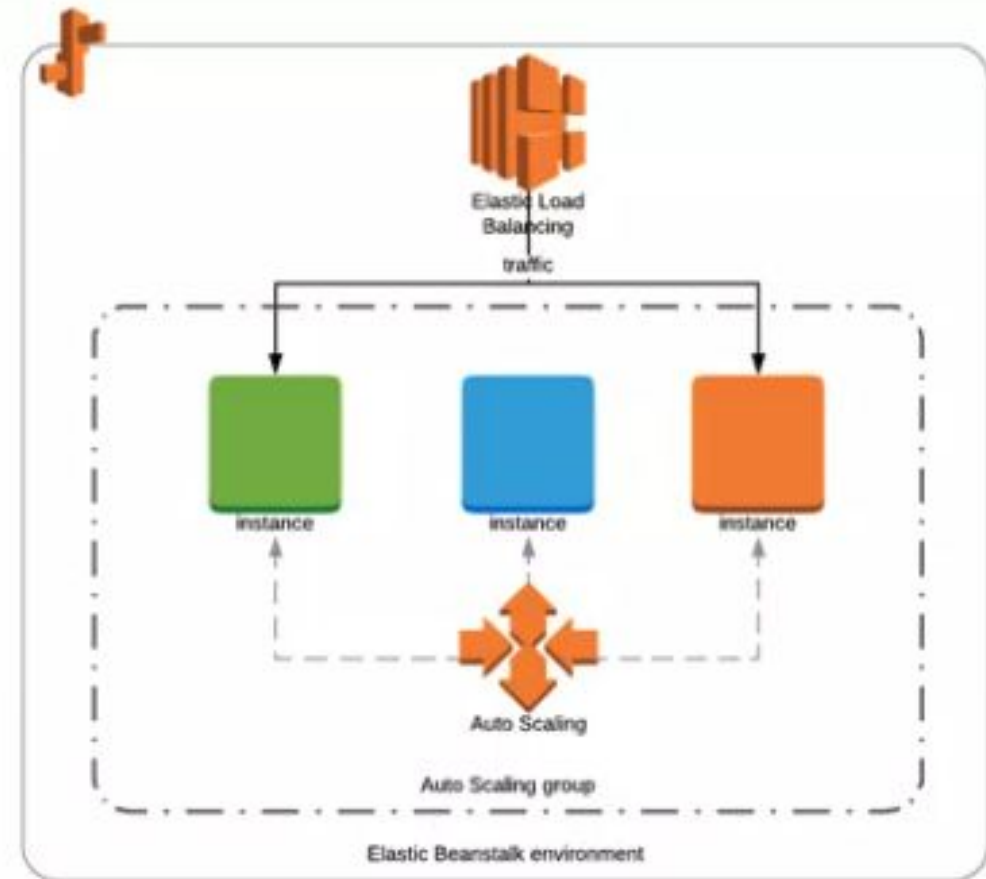


2. In case of failure, all fails!

# Deployment Models - Rolling

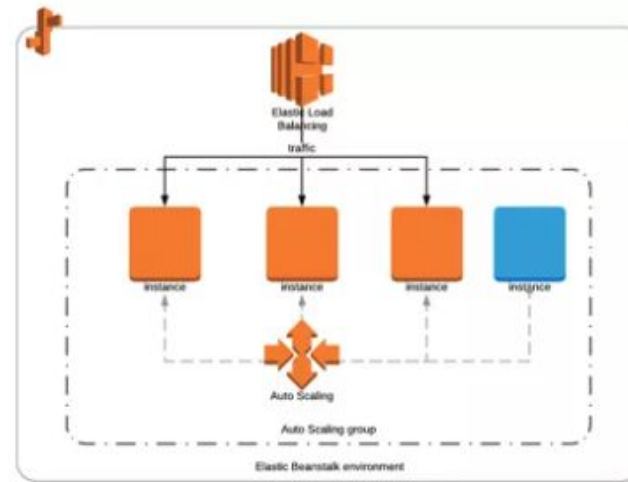


1. Deployment starts in the first batch.

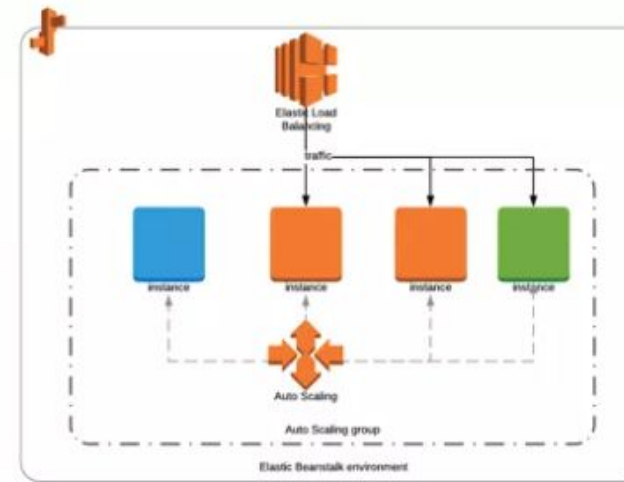


2. Deployment continues with the next batch.

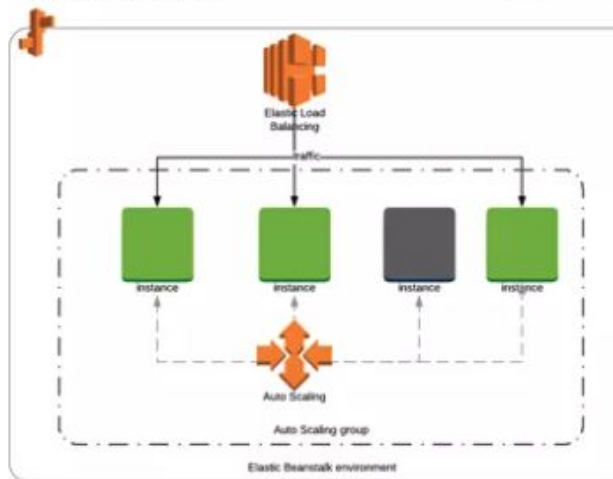
# Deployment Models - Rolling with additional batch



1. Deployment starts by launching new instances for the first batch.

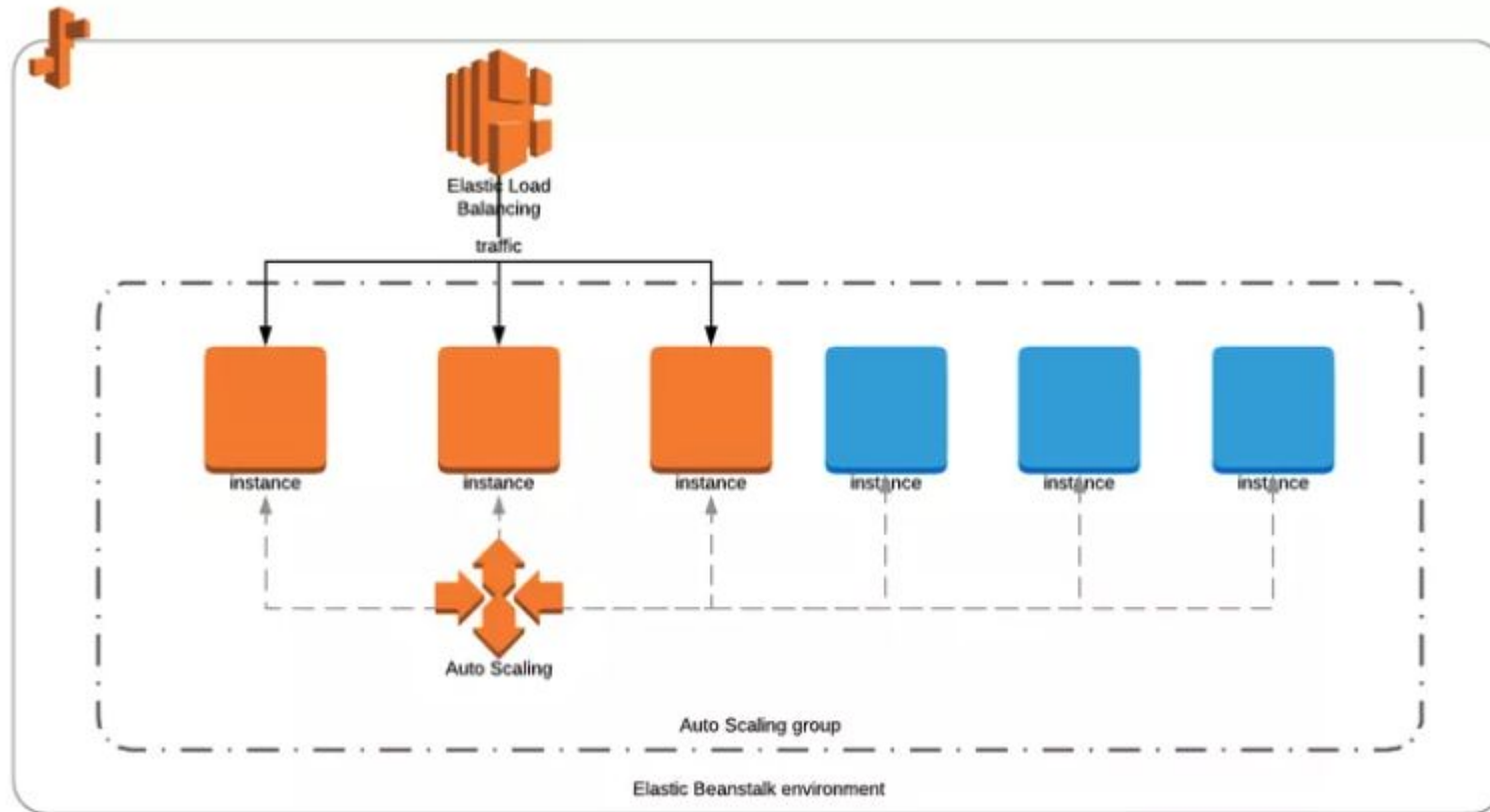


2. Deployment continues with the next batch.



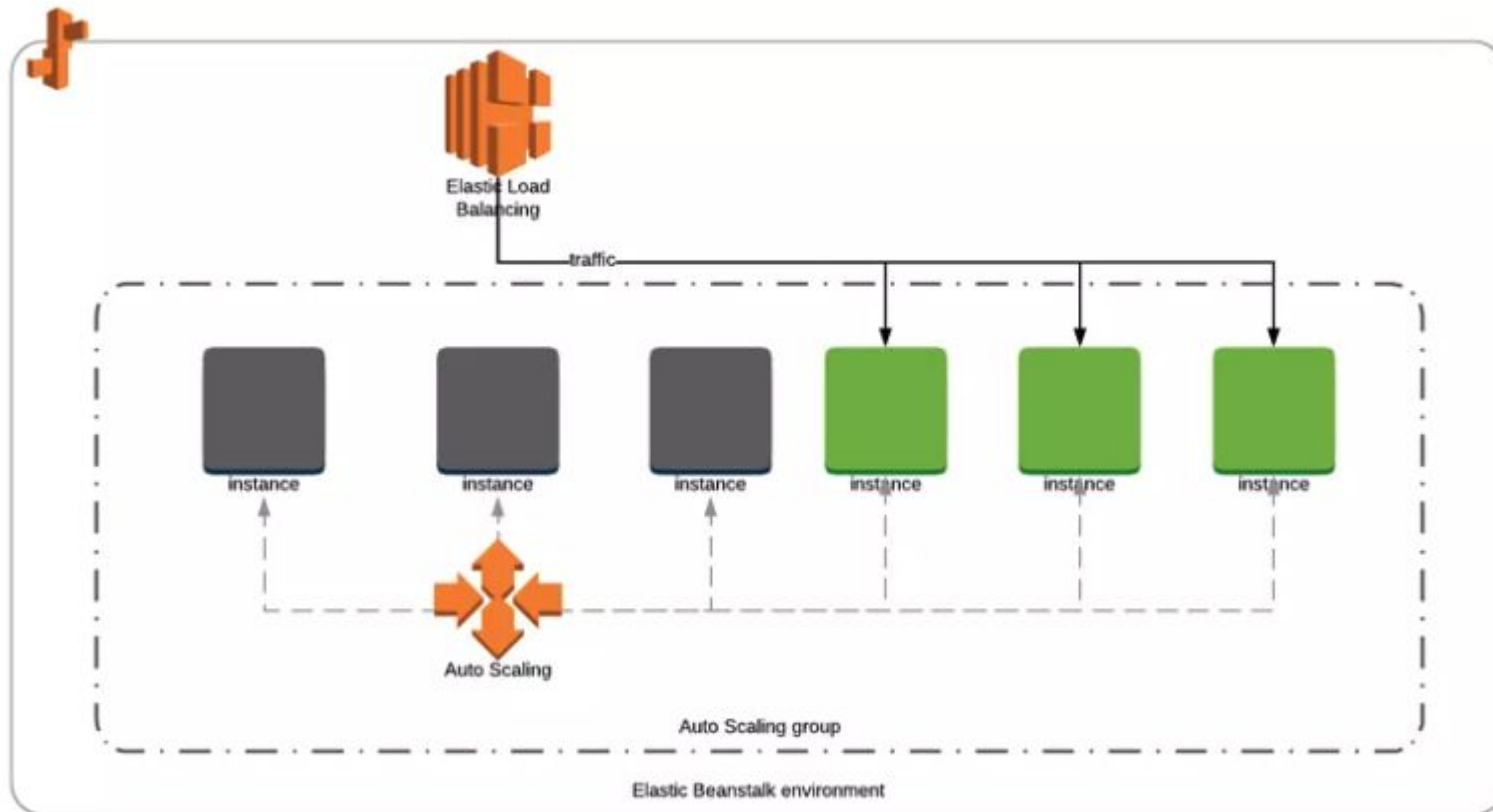
3. After the final batch, excess instance is terminated.

# Deployment Models - Immutable



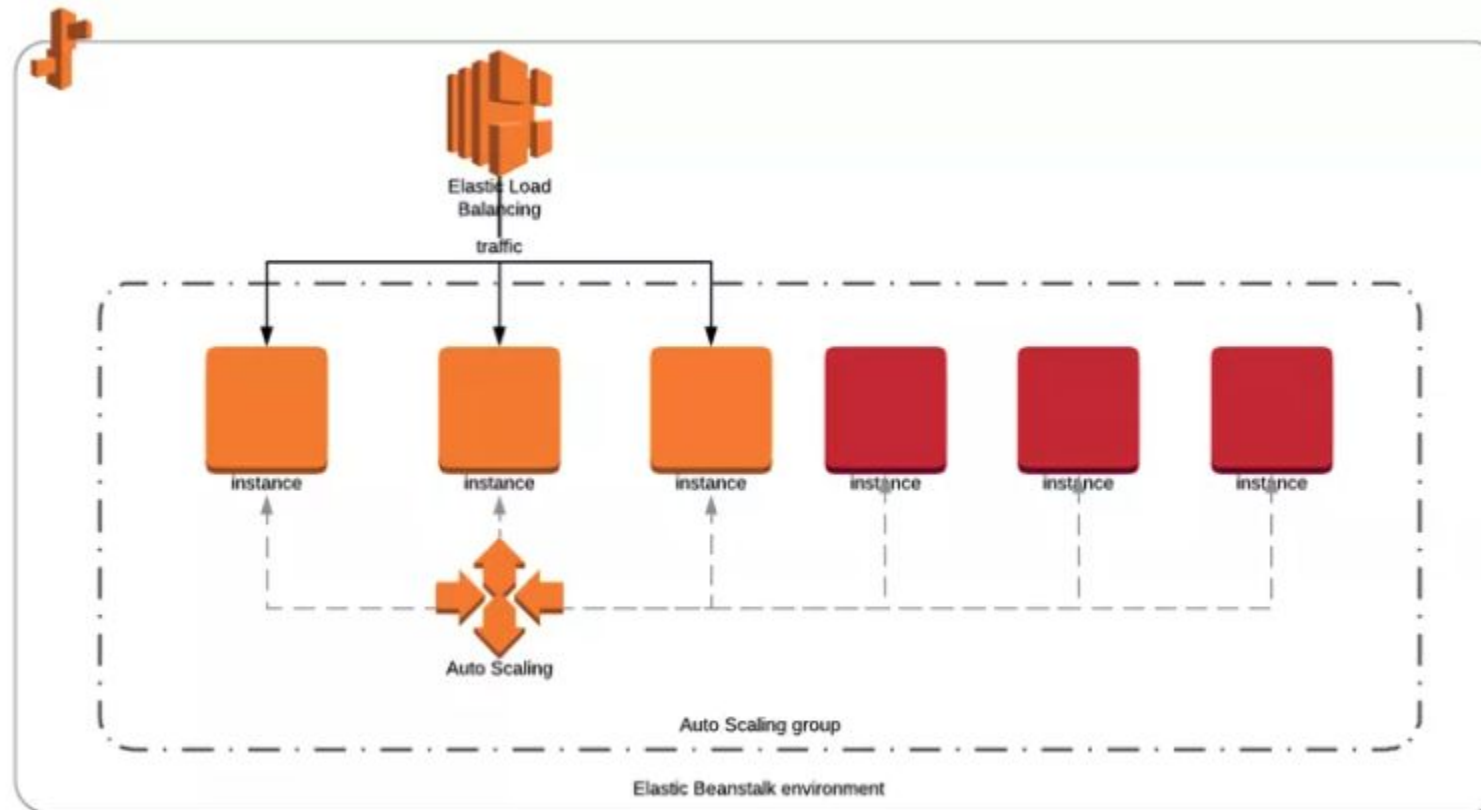
1. Deployment starts by duplicating the instances and deploying the app in new instances

# Deployment Models - Immutable



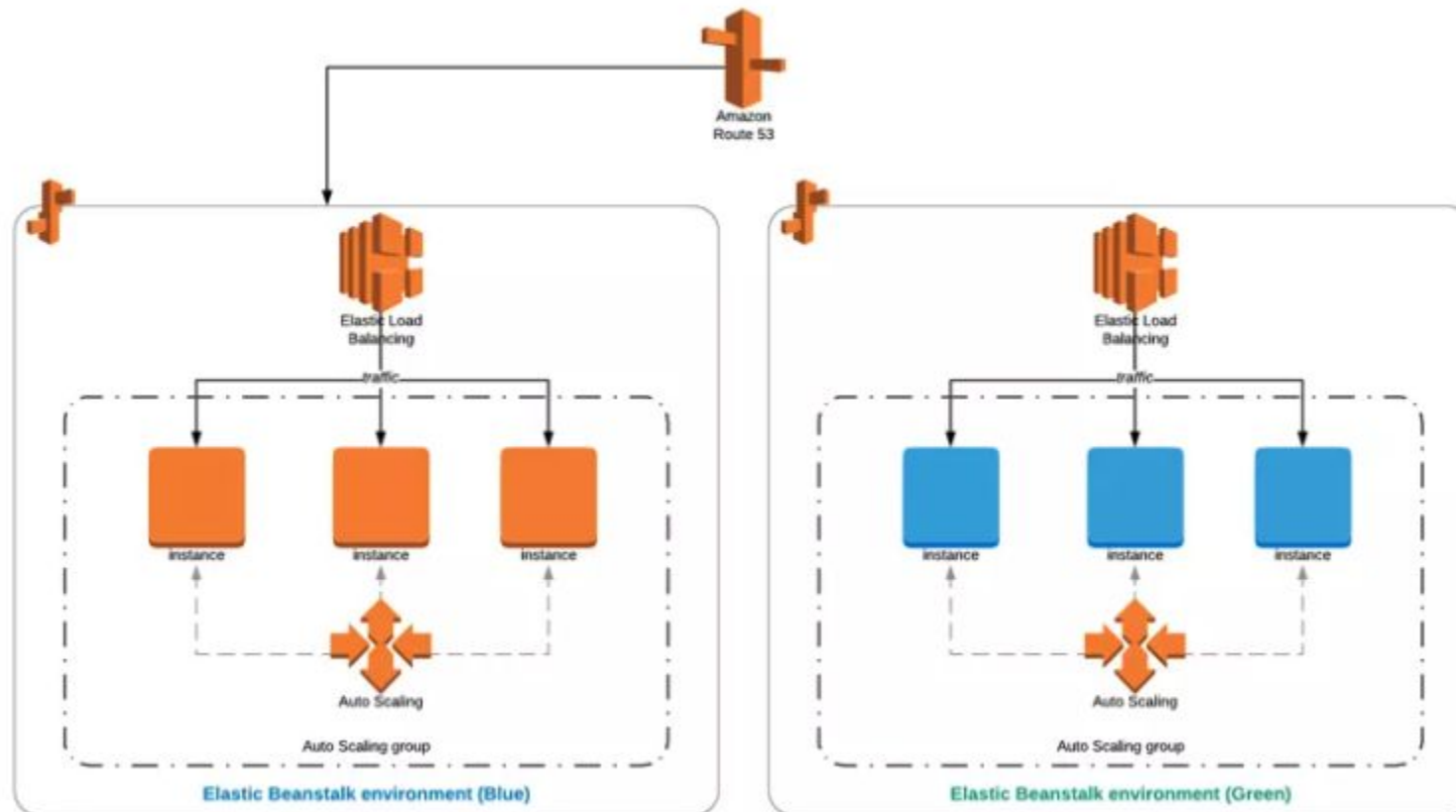
2. After deployment succeeds, older instances are terminated.

# Deployment Models - Immutable



In case of failure, the new instances is terminated and traffic is not effected.

# Deployment Models - Blue/Green

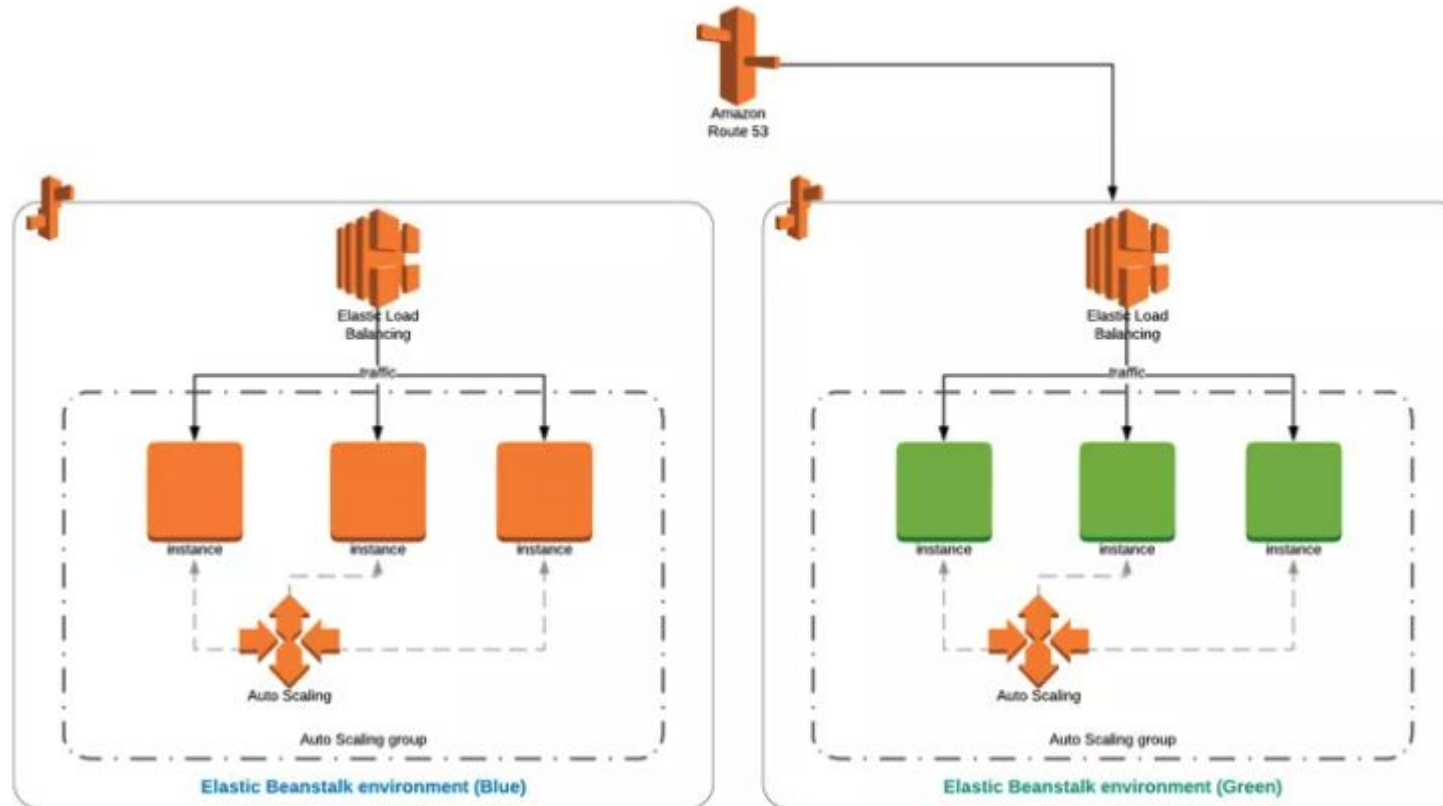


Old environment continues to serve the traffic during deployments on the cloned environment. In case of failure, it will not be effected.

Deployments are run on the cloned environment



# Deployment Models - Blue/Green



# Deployment Models - Pros & Cons



Supported deployment policies			
Deployment policy	Load-balanced environments	Single-instance environments	Legacy Windows Server environments†
All at once	✓ Yes	✓ Yes	✓ Yes
Rolling	✓ Yes	✗ No	✓ Yes
Rolling with an additional batch	✓ Yes	✗ No	✗ No
Immutable	✓ Yes	✓ Yes	✗ No
Traffic splitting	✓ Yes (Application Load Balancer)	✗ No	✗ No



Let's get our hands dirty!

- Creating Application



# THANKS!

**Any questions?**

