WELCOME

DevOps: Continuous Integration (CI) & Continuous Deployment (CD) using AWS CodePipeline & AWS Elastic Beanstalk

AWS CodePipeline

AWS Elastic Beanstalk

INTRODUCTION

- Fast-paced, extremely competitive landscape
- Emergence of cloud-based infrastructure
- **Time to launch** new features has become critical
- Roles such as **DevOps**
- **Automation and CI/CD** are crucial for a successful DevOps team

Learn to build **fully automated** CI/CD pipelines for web applications on **Amazon's Cloud.**

Why?

- DevOps skills (such as CI/CD) are highlysought after
- AWS is the leading cloud service provider
- Get an in-depth hands-on experience through this course

WHY THIS COURSE?

LET'S START!

Before we start...

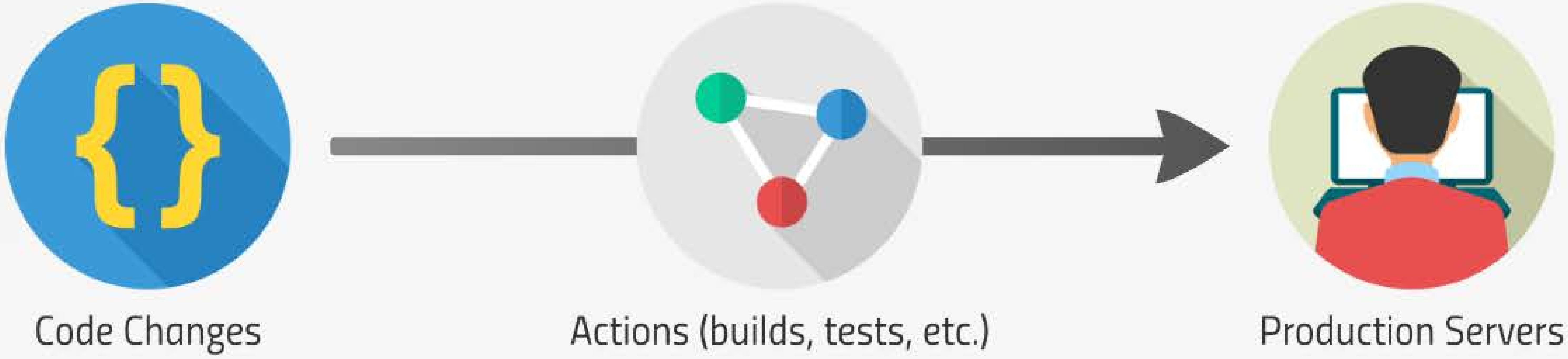
- Pre-requisites (Terminal, Software Development, etc.)
- Amazon Web Services (AWS)

COURSE OUTLINE

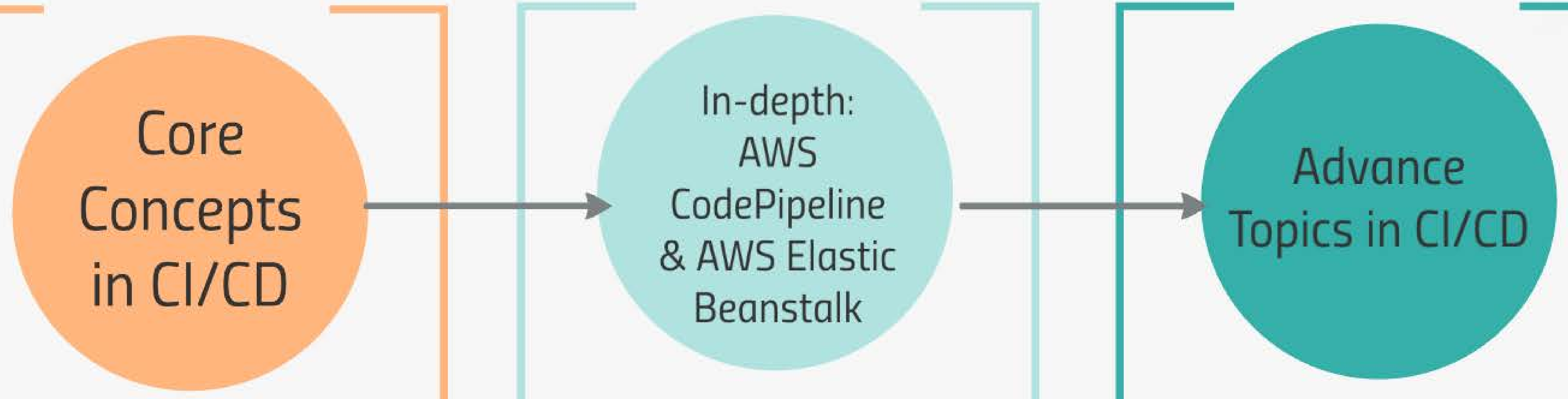
- How is the course structured?
- What will you learn?
- What will we build together in our workshops?

VERY QUICK CI/CD OVERVIEW

- **Continuous Integration:** Code changes are built, prepared and tested automatically.
- **Continuous Deployment:** Expands upon CI by deploying those changes out to testing or production servers.



HOW IS THE COURSE STRUCTURED?



Core Concepts in CI/CD

- Basic concepts
- Fully automated CI/CD Pipelines
- CI/CD for web applications
- Common challenges

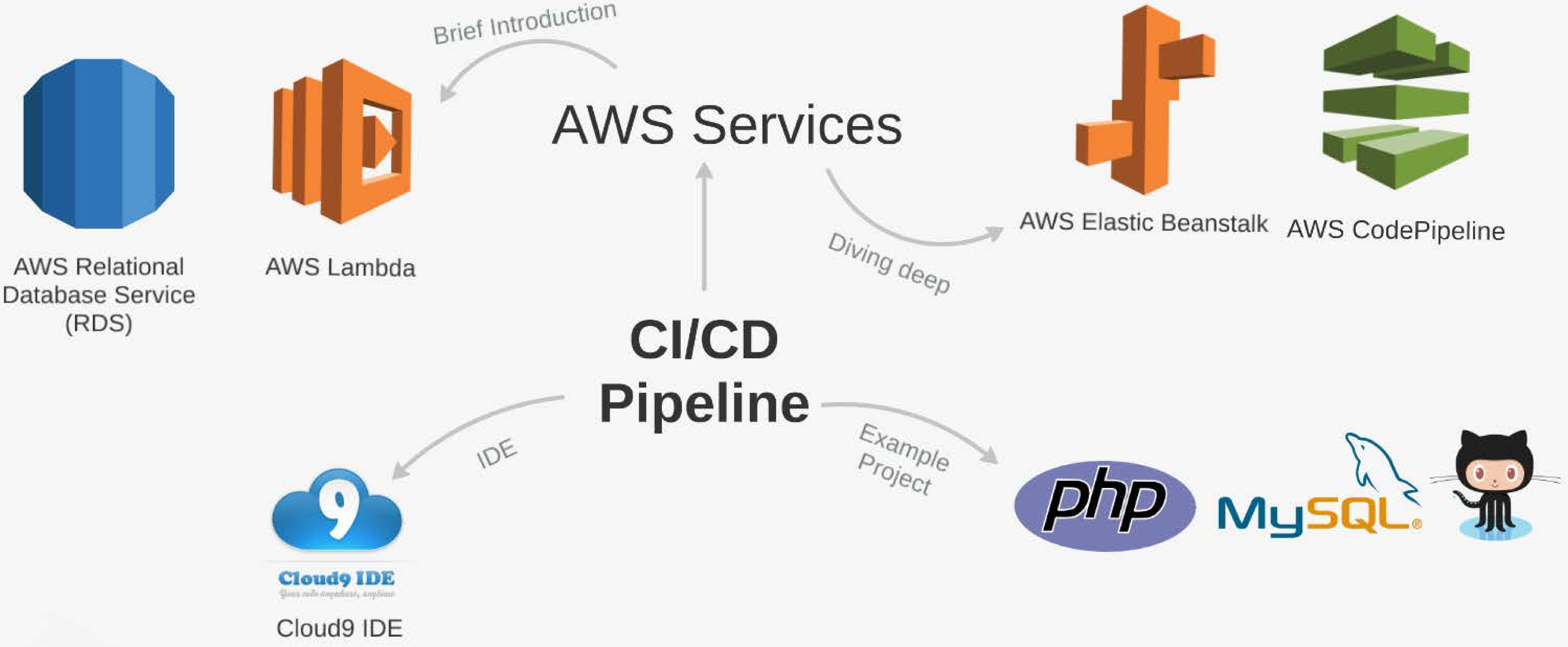
In-depth: AWS CodePipeline & AWS Elastic Beanstalk

- Basic PHP web application
- Write and running tests
- Database Management

- Deploy using AWS Elastic Beanstalk
- Setup MySQL database on AWS RDS
- Manual CI/CD AWS CodePipeline

Advance Topics in CI/CD

- AWS CodePipeline Actions
    - Example AWS Lambda Action
- AWS Elastic Beanstalk Extensions
- Setup a fully automated pipeline
- Monitoring the pipeline
- Real-life scenarios & examples



CI/CD Pipeline

AWS Services

Brief Introduction

AWS Relational Database Service (RDS)

AWS Lambda

Diving deep

AWS Elastic Beanstalk

AWS CodePipeline

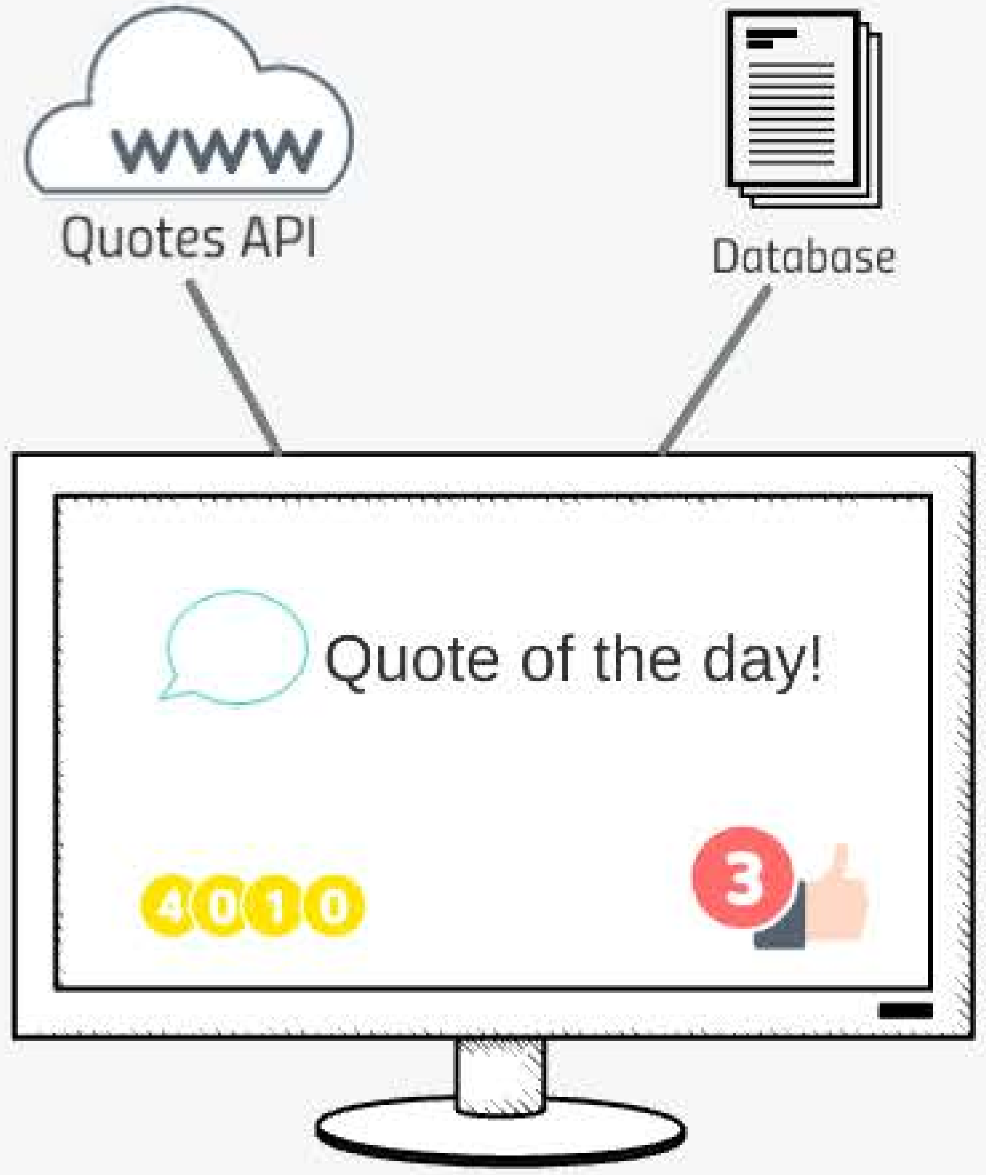
IDE

Cloud9 IDE

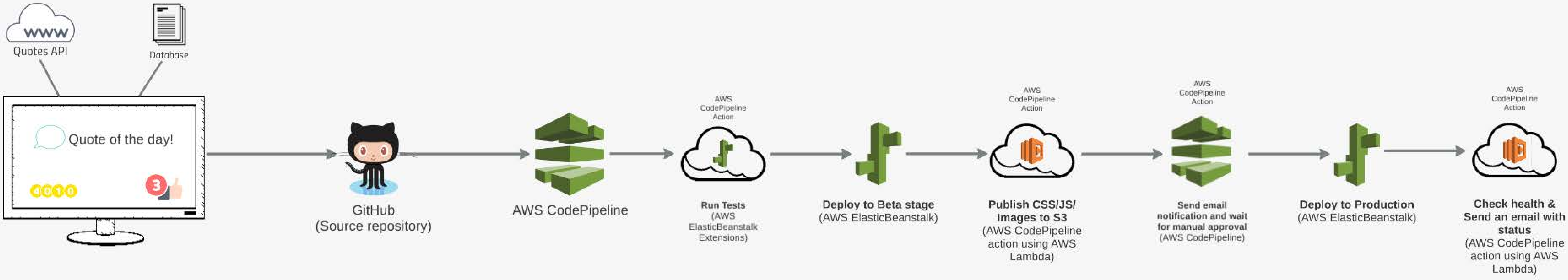
Example Project

WHAT WILL YOU LEARN?

WHAT WILL WE BUILD?



WHAT WILL WE BUILD?



Quotes API

Database

GitHub
(Source repository)

AWS CodePipeline

AWS CodePipeline Action
Run Tests
(AWS ElasticBeanstalk Extensions)

Deploy to Beta stage
(AWS ElasticBeanstalk)

AWS CodePipeline Action
Publish CSS/JS/ Images to S3
(AWS CodePipeline action using AWS Lambda)

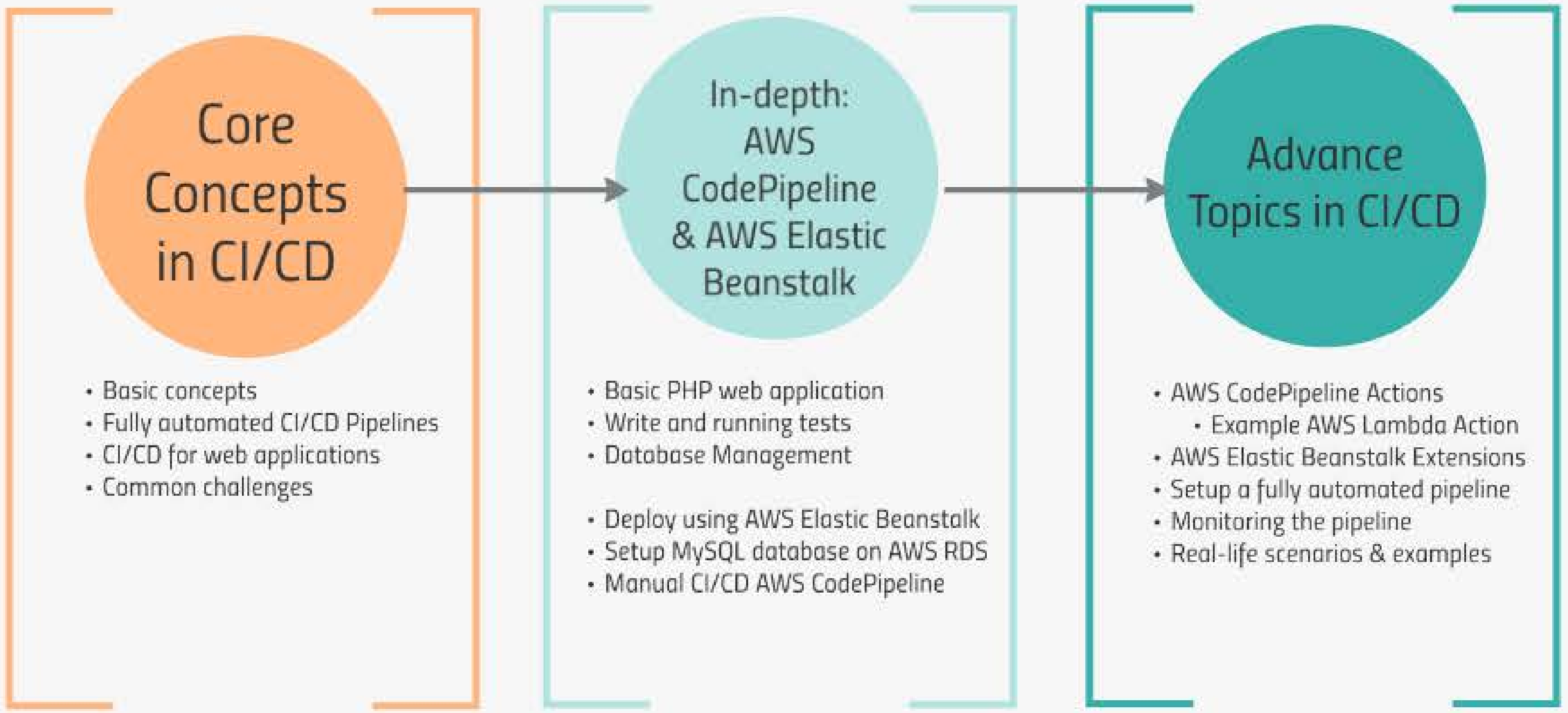
AWS CodePipeline Action
Send email notification and wait for manual approval
(AWS CodePipeline)

Deploy to Production
(AWS ElasticBeanstalk)

AWS CodePipeline Action
Check health & Send an email with status
(AWS CodePipeline action using AWS Lambda)

SECTION OVERVIEW

Introduction to Continuous Integration & Continuous Deployment



Core Concepts in CI/CD

- Basic concepts
- Fully automated CI/CD Pipelines
- CI/CD for web applications
- Common challenges

In-depth: AWS CodePipeline & AWS Elastic Beanstalk

- Basic PHP web application
- Write and running tests
- Database Management

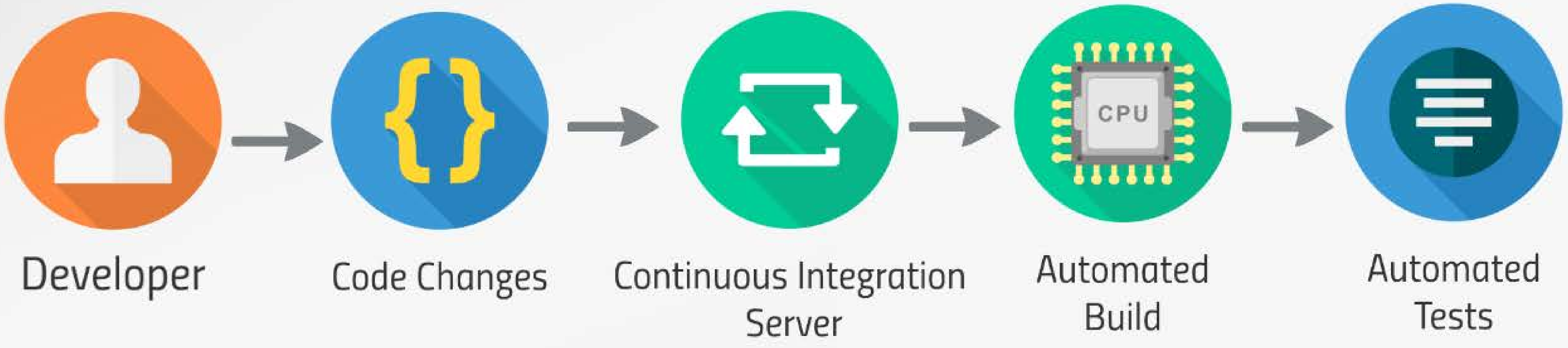
- Deploy using AWS Elastic Beanstalk
- Setup MySQL database on AWS RDS
- Manual CI/CD AWS CodePipeline

Advance Topics in CI/CD

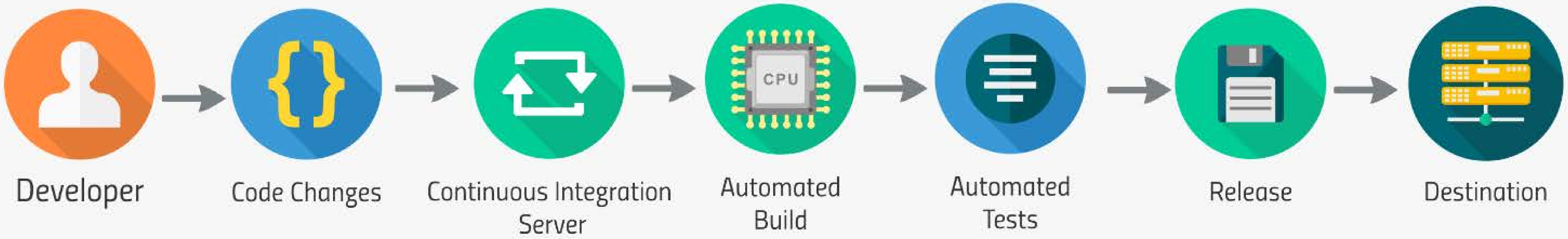
- AWS CodePipeline Actions
  - Example AWS Lambda Action
- AWS Elastic Beanstalk Extensions
- Setup a fully automated pipeline
- Monitoring the pipeline
- Real-life scenarios & examples

WHAT IS CI?

Continuous Integration means **regularly** and frequently **merging and building changes** to your software.

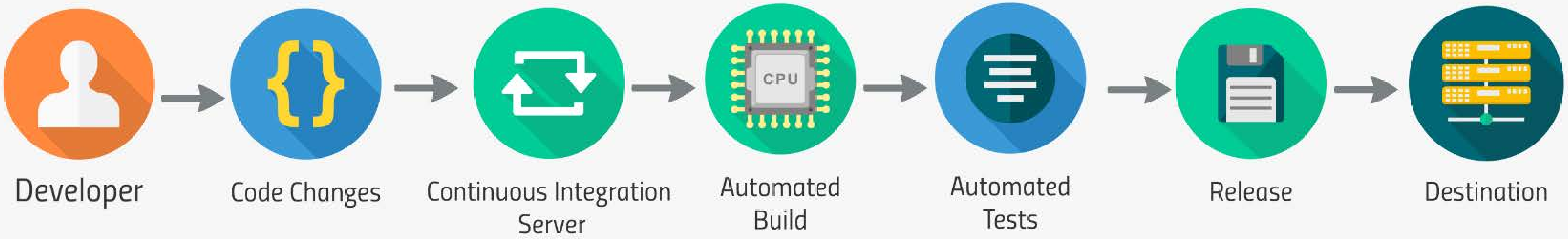


- **Continuous Delivery** means **regularly** releasing your software to a **destination**.
- **Continuous Deployment:** Release goes to customers/end-users (e.g. to production).



WHAT IS CD?

- **Pipeline:** Sequence of steps a code change goes through.
- Depending on the steps a pipeline can either be CI or CD.



WHAT IS CI/CD PIPELINE?

inuous Integration Server

Automated Build

Automated Tests

Release

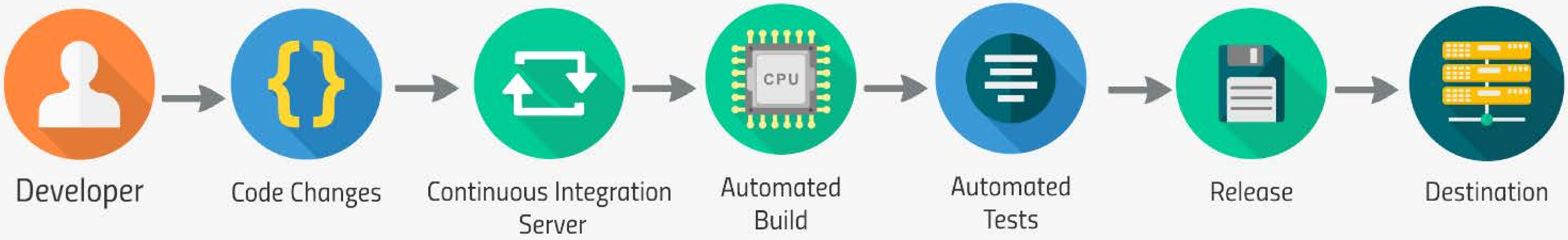
Destination

CI/CD PIPELINE?

Pipeline Building Blocks

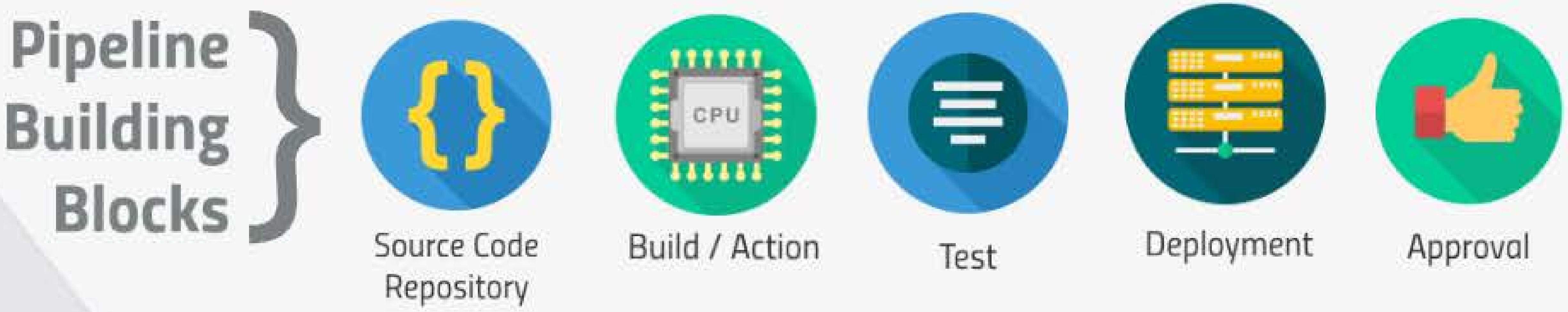
- Source Code Repository
- Build / Action
- Test
- Deployment
- Approval

- **Pipeline:** Sequence of steps a code change goes through.
- Depending on the steps a pipeline can either be CI or CD.



Developer → Code Changes → Continuous Integration Server → Automated Build → Automated Tests → Release → Destination

WHAT IS CI/CD PIPELINE?



Pipeline Building Blocks

Source Code Repository

Build / Action

Test

Deployment

Approval

WHY USE CI/CD?

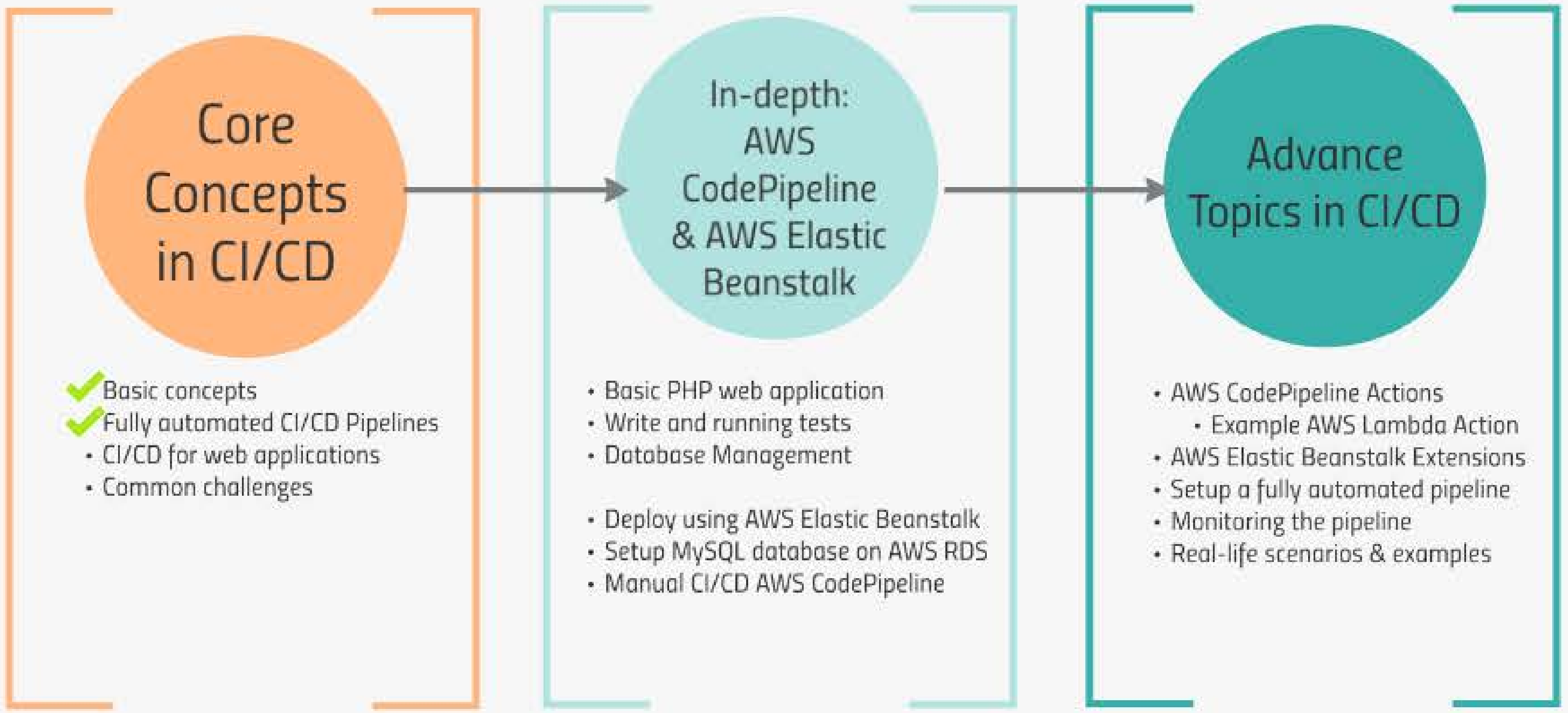
- **Moving Fast:** Release changes and updates frequently
- **Reliability:** Consistent process for each and every change.
- **Rapid Troubleshooting:** A test failure will block the pipeline and merits immediate investigation.
- **Avoid Stress:** Takes the stress away from scrambling to deploy emergency fixes or updates.

FULLY AUTOMATED PIPELINE

- **Fully Automated:** CD Pipeline with no manual approval step.
- **Problem:** Issues and bugs shipped to customers.
- **Solution:** Automated Detection (Monitoring and Logging).
- **Even Better:** Automated Recovery (Rollbacks).

SECTION OVERVIEW

Continuous Integration & Deployment for Web Applications



Core Concepts in CI/CD

- ✓ Basic concepts
- ✓ Fully automated CI/CD Pipelines
- CI/CD for web applications
- Common challenges

In-depth: AWS CodePipeline & AWS Elastic Beanstalk

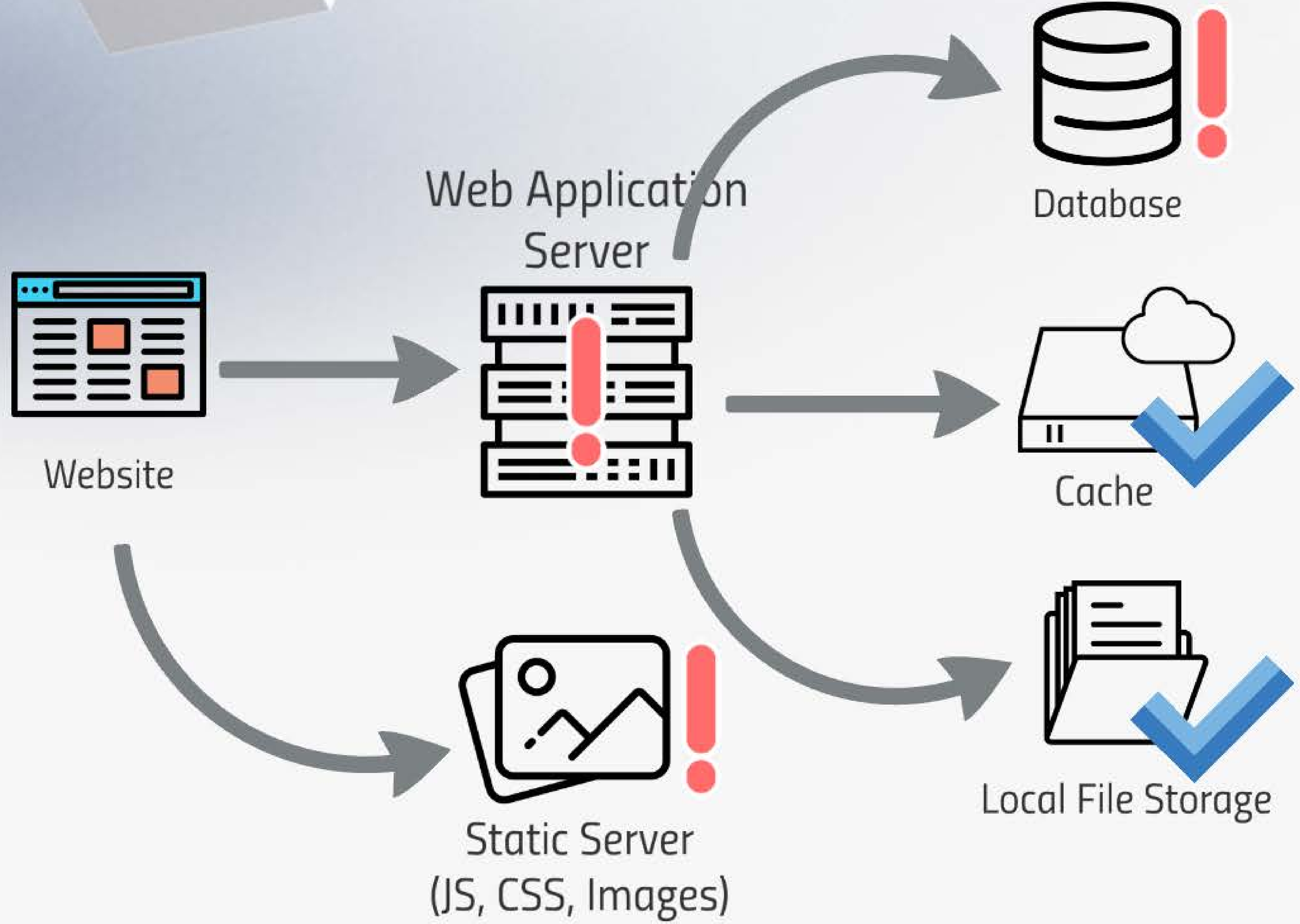
- Basic PHP web application
- Write and running tests
- Database Management

- Deploy using AWS Elastic Beanstalk
- Setup MySQL database on AWS RDS
- Manual CI/CD AWS CodePipeline

Advance Topics in CI/CD

- AWS CodePipeline Actions
  - Example AWS Lambda Action
- AWS Elastic Beanstalk Extensions
- Setup a fully automated pipeline
- Monitoring the pipeline
- Real-life scenarios & examples

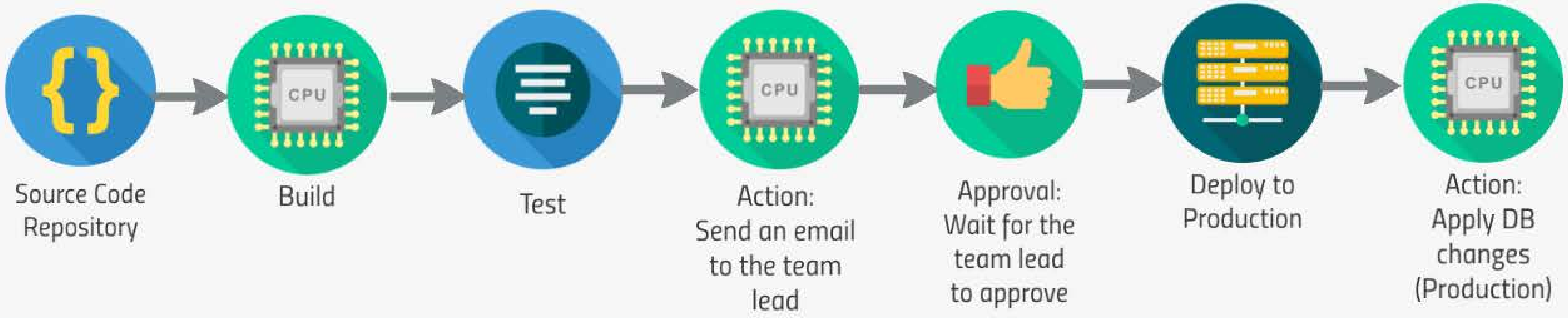
BASIC WEB APPLICATION



1. What are the most common changes?
2. As a result, which components will you manually update?
3. Which CD Pipeline Building Block will you use?

4 Model your process → Run tests   Send an email to the team lead

Manual approval   Deploy to Production


Source Code Repository
Build
Test
Action: Send an email to the team lead
Approval: Wait for the team lead to approve
Deploy to Production
Action: Apply DB changes (Production)

CD PIPELINE FOR WEB APPLICATIONS

MANAGING DATABASE CHANGES

Issues:

- Database changes are **not** done in code or stored as code.
- Changes to data (e.g. data stored in production database) are not replicated in code repository.

Observation:

Changes are either schema changes or initialization data.

Solution:

Treat database changes as code.

Types of Actions

Standalone action

Runs Independently.
Doesn't require executing the application.
Doesn't rely on the application code.
Runs as part of the CI/CD pipeline.

Deployment action

Requires an active deployment to run.
May require libraries in the application or rely on the application code itself.
Triggered by CI/CD pipeline but runs during the deployment process.

Action

Deployment

DEPLOYMENT & ACTIONS

MONITORING

Automated Pipelines need automated failure detection.

- **HTTP Ping:** Get a successful HTTP response (200).
- **Deep HTTP Ping:** Keywords are present in the HTTP response.
- **Functional Checks:** Simulate interactions and validate results.
- **Metrics:** Host and application metrics.

COMMON CHALLENGES

- **Deploying changes while serving traffic:**
  - Batch changes for production.
  - Apply changes to a standby environments and then flip.
  - Apply time-based approval window.
- **Application caches / cached files:**
  - Use Cache buster tags.
  - Use versioned files.