# Copyright Notice

DeepLearning.AI

# Source Systems, Data Ingestion, and Pipelines

## Week 3

DeepLearning.AI

# DataOps

## Week 3 Overview

# DataOps

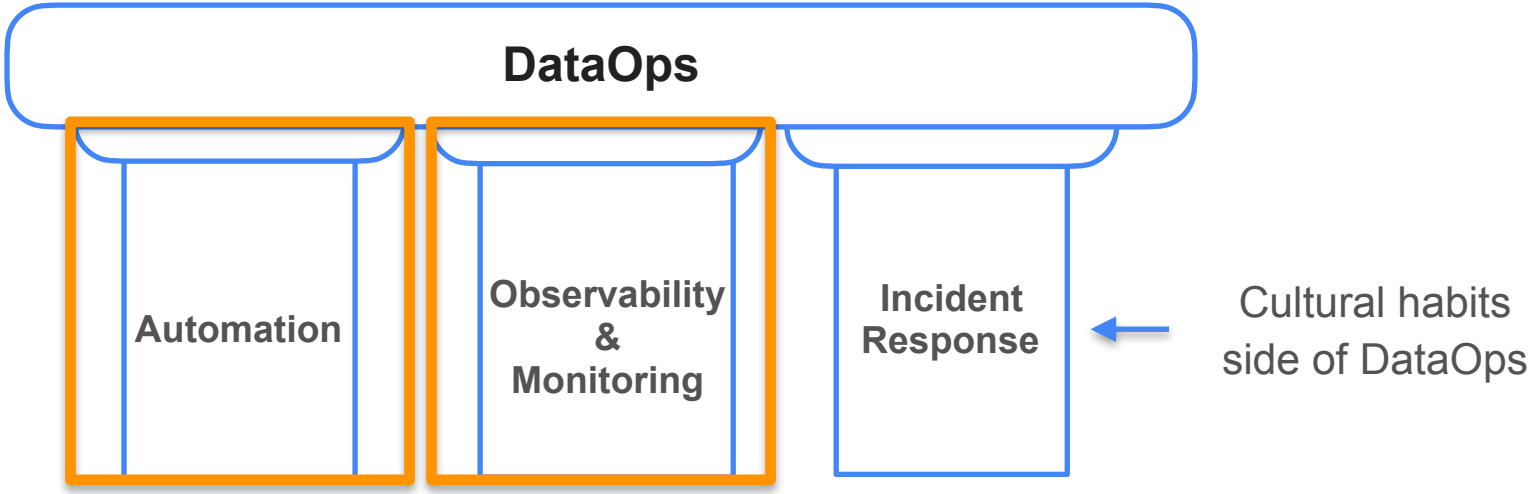
**DataOps**

Set of practices and cultural habits centered around building robust data systems and delivering high quality data products.

**DevOps**

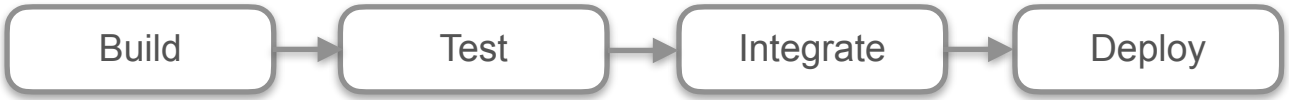
Set of practices and cultural habits that allow software engineers to efficiently deliver and maintain high quality software products.

# The Three Pillars of DataOps



# Automation

**Continuous Integration and Continuous Delivery (CI/CD)**




**Infrastructure as code**

Code that acts to deploy the resources required to run your data pipeline

AWS CloudFormation

# Week 3 Labs

HashiCorp Terraform

Hands-on experience creating resources using Terraform

Monitoring Data Quality & other Observability Metrics

# Interviews with Industry Experts

Chris Bergh

Barr Moses

Abe Gong

Chad Sanderson

DataKitchen

MC MONTE CARLO

great expectations

gable

DeepLearning.AI

# DataOps

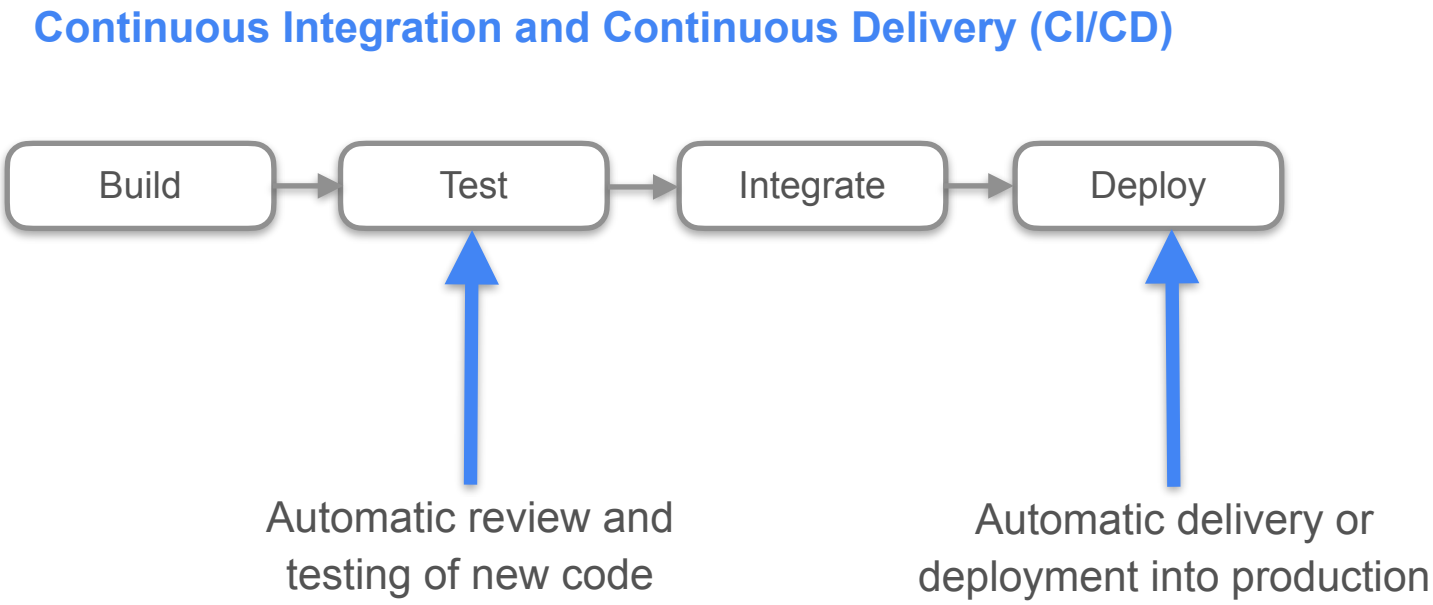
**Chris Bergh**
**CEO at DataKitchen**

DeepLearning.AI

# DataOps - Automation

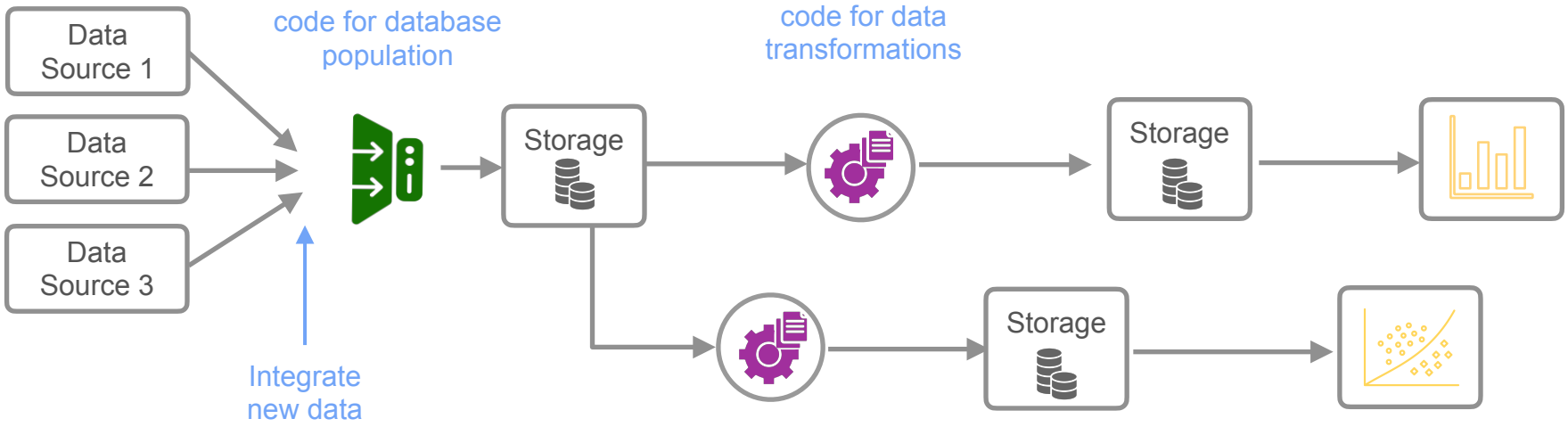
## DataOps Automation

# DevOps Automation



# DataOps Automation

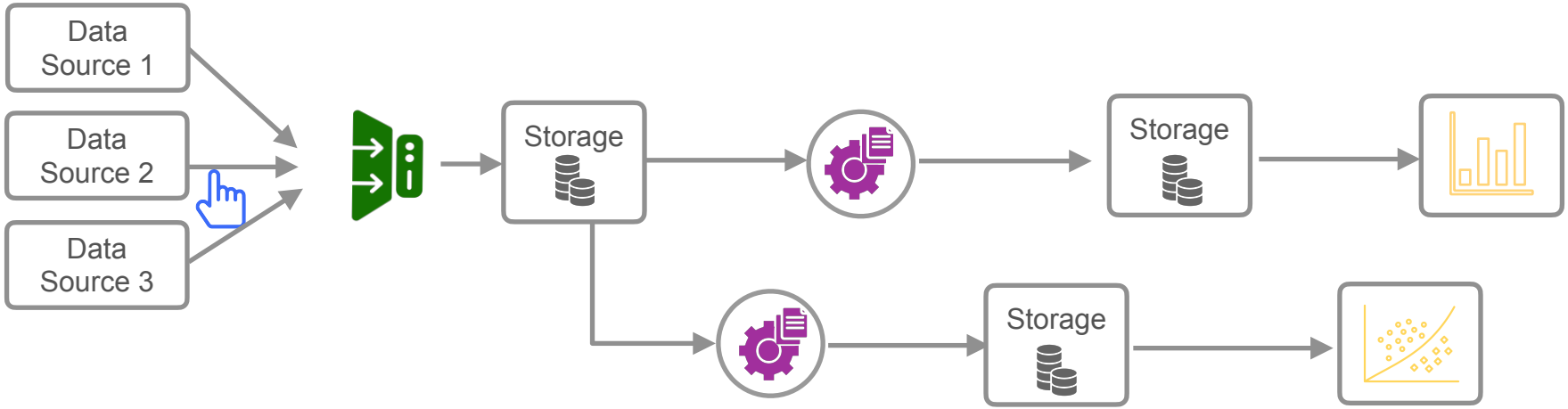
CI/CD can be applied directly to code and data within your data pipeline



# DataOps Automation

**Automation of data pipeline running**

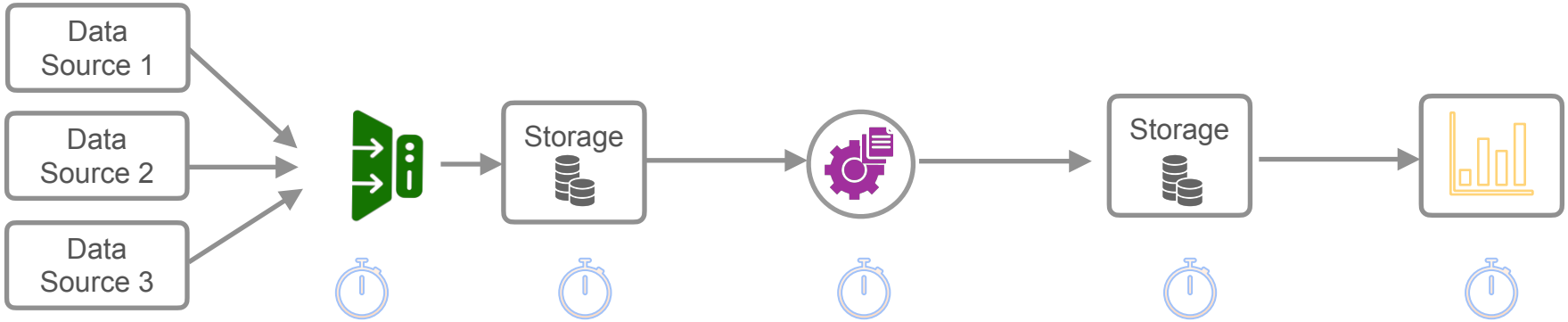
**No automation:** run all processes manually



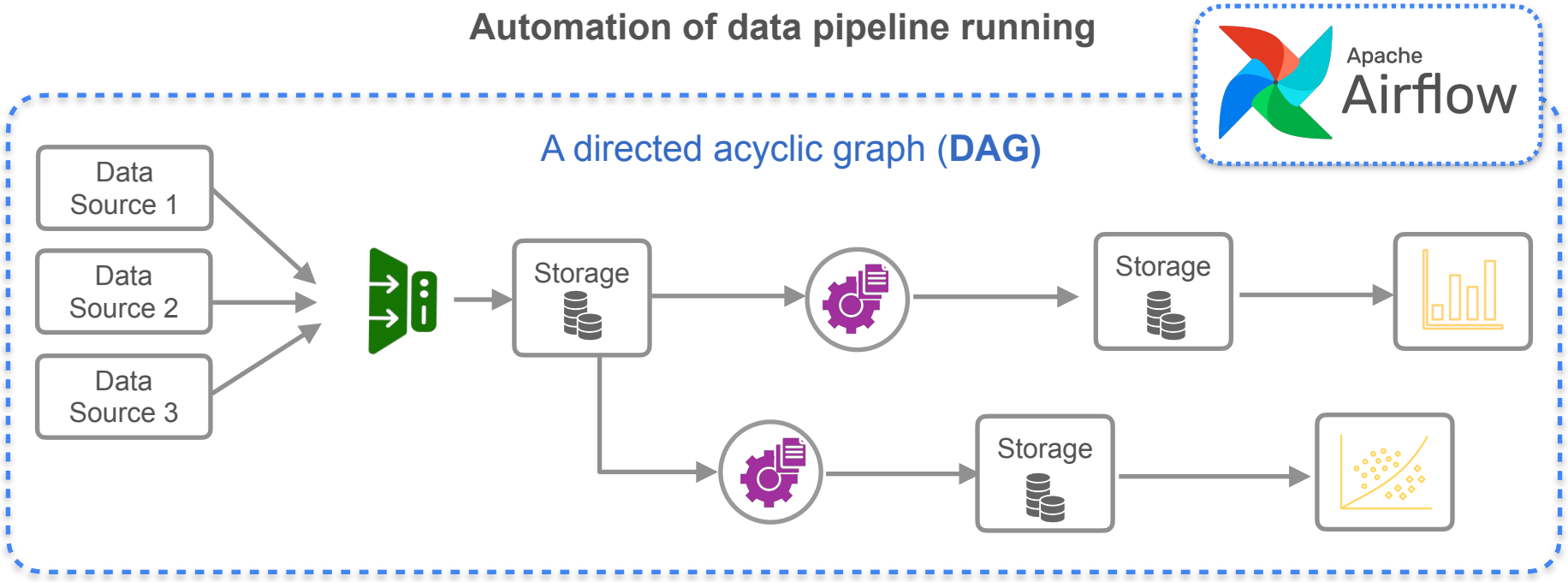
# DataOps Automation

**Automation of data pipeline running**

**Pure scheduling:** run stages of your pipeline according to a schedule



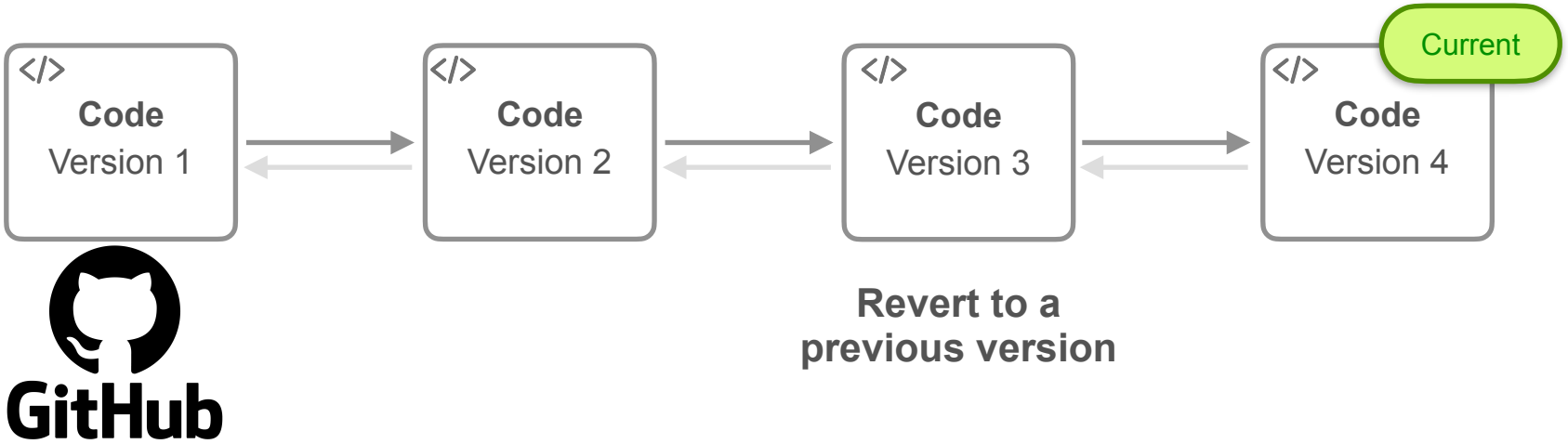
# DataOps Automation



# DataOps Automation

**Key Underpinning CI/CD : Version control**

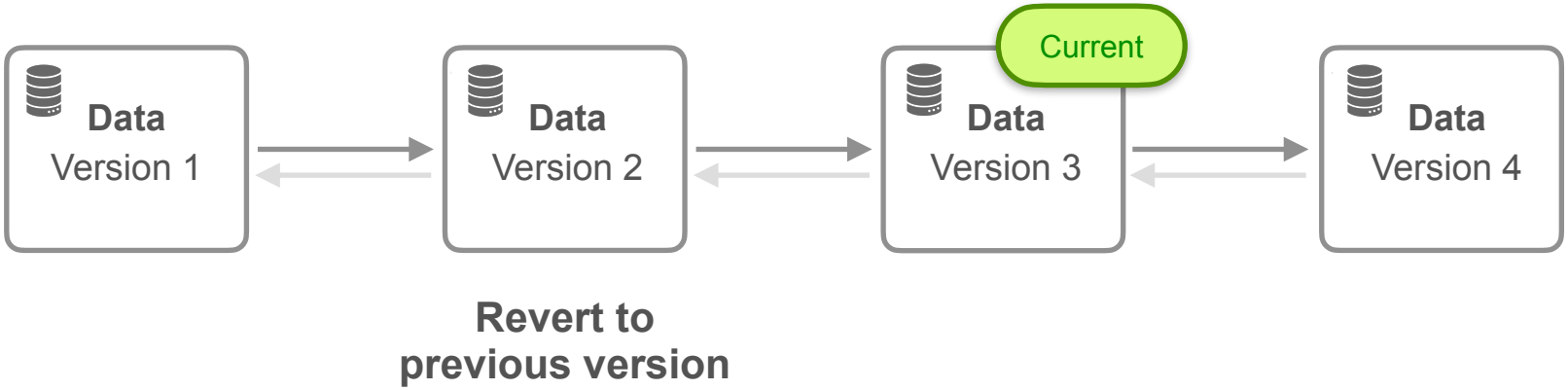
Track changes in the **code**



# DataOps Automation

**Key Underpinning CI/CD : Version control**

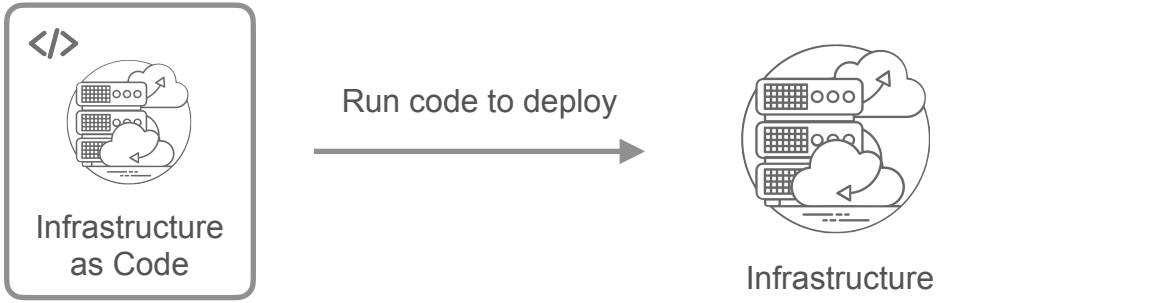
Track changes in the **code** and the **data**



# DataOps Automation

**Key Automation Aspect: Infrastructure as Code**

Maintain the design of your infrastructure as a codebase

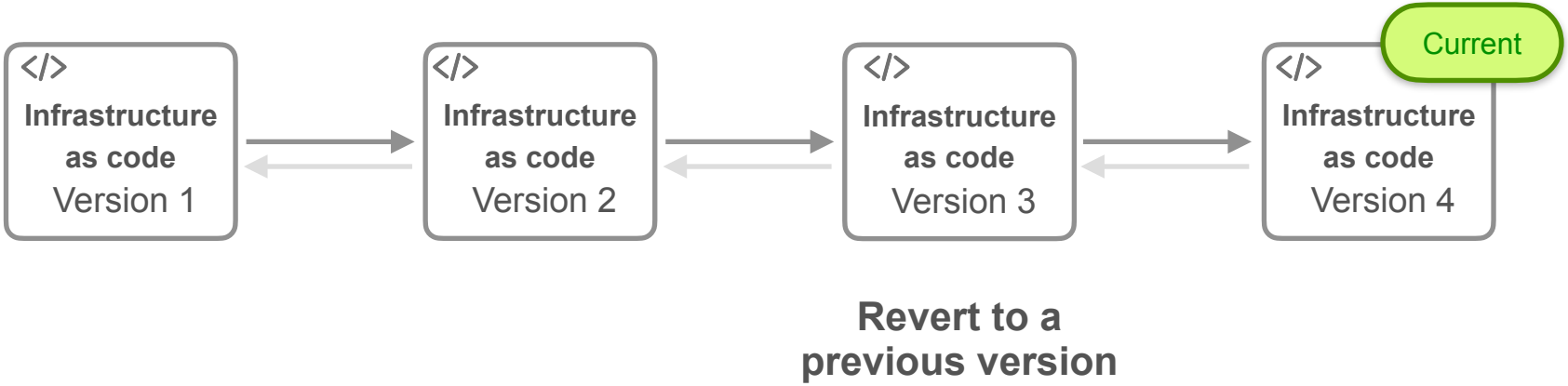



Modify this code to redefine your infrastructure

# DataOps Automation

**Version control over your entire infrastructure**

Track changes in the **code** defining your infrastructure



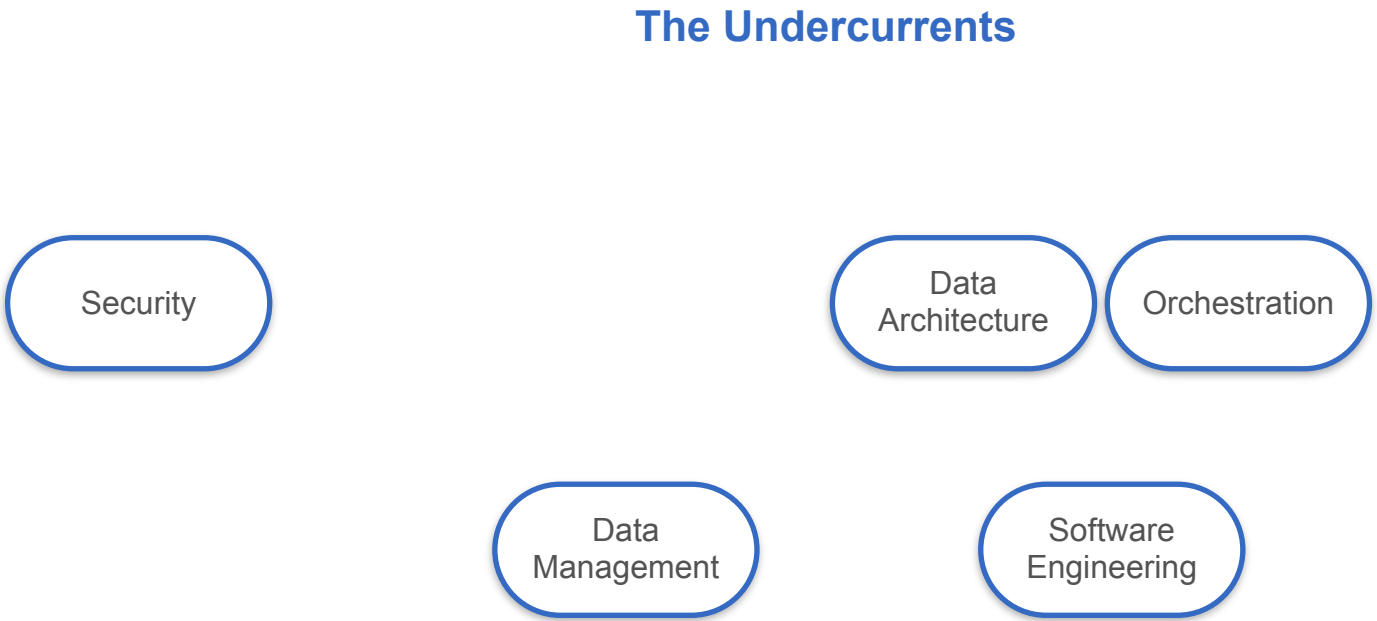
# DataOps

## The Undercurrents

# DataOps

## The Undercurrents

# DataOps


The Undercurrents
Security
Data Architecture
Orchestration
Data Management
Software Engineering

DeepLearning.AI

# DataOps - Automation

## Infrastructure as Code

# Infrastructure as Code

**1970s**

**Configuration Management**

**Challenge :** configuration of a series of physical machines

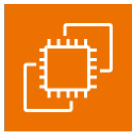

To automate some configuration tasks

# Infrastructure as Code

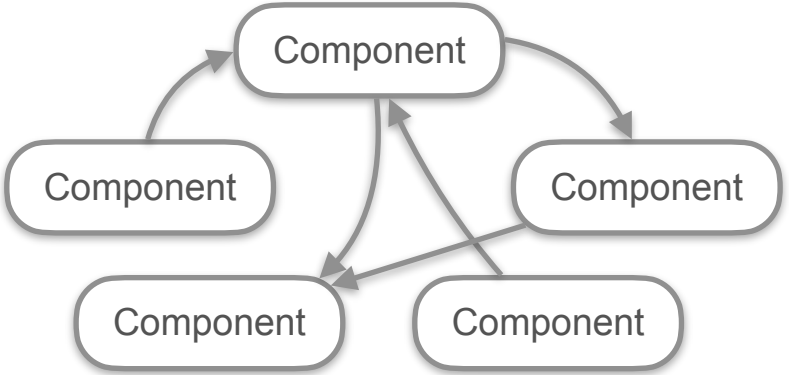
**1970s**

Configuration Management

**2006**



Amazon Elastic Compute Cloud (Amazon EC2)



# Infrastructure as Code

**1970s**

Configuration Management

**2006**

Amazon Elastic Compute Cloud (Amazon EC2)

**2010s**

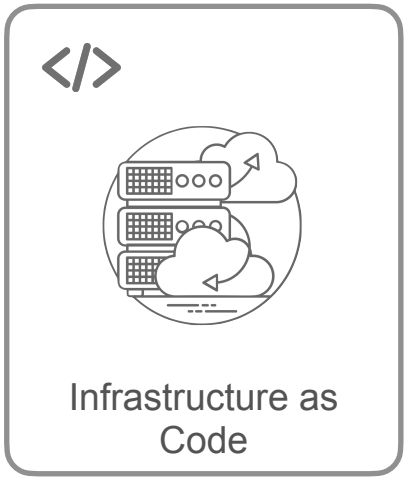
**Development of Infrastructure as Code Tools**

HashiCorp Terraform

AWS CloudFormation

ANSIBLE

# Infrastructure as Code



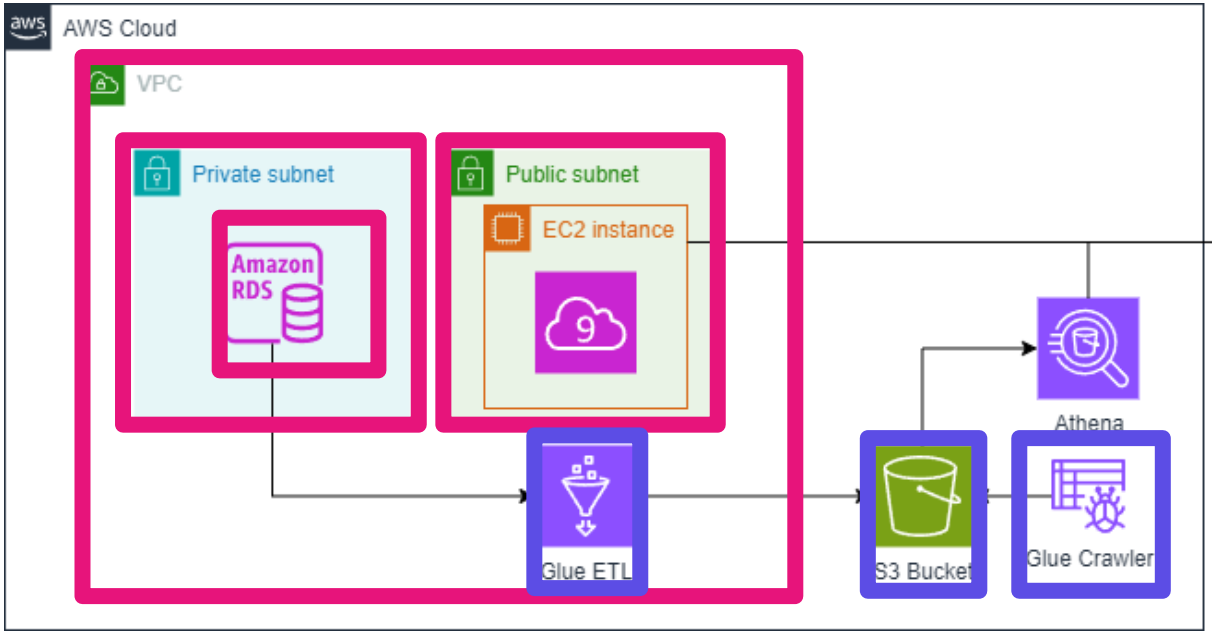

AWS CloudFormation

HashiCorp Terraform

ANSIBLE

- Manage infrastructure resources on the cloud using code
- No manual clicking or writing bash scripts

# Course 1 Week 2 Lab




AWS CloudFormation

HashiCorp Terraform

# Infrastructure as Code Tools

HashiCorp Terraform

- Used in the lab
- Well supported with great documentation

AWS CloudFormation

- Native to AWS
- Well supported with great documentation

# Terraform Configuration Files (C1W2 Lab)

```
resource "aws_s3_bucket" "data_lake" {
  bucket_prefix = "${var.project}-datalake-${data.aws_caller_identity.current.account_id}-"
}

resource "aws_s3_bucket_public_access_block" "data_lake" {
  bucket = aws_s3_bucket.data_lake.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_s3_bucket" "scripts" {
  bucket_prefix = "${var.project}-scripts-${data.aws_caller_identity.current.account_id}-"
}

resource "aws_s3_bucket_public_access_block" "scripts" {
  bucket = aws_s3_bucket.scripts.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_s3_object" "glue_job_script" {
  bucket = aws_s3_bucket.scripts.id
```

```
resource "aws_glue_catalog_database" "analytics_database" {
  name        = "${var.project}-analytics-db"
  description = "Database for performing analytics on OLTP data"
}

resource "aws_glue_connection" "rds_connection" {
  name = "${var.project}-rds-connection"

  connection_properties = {
    JDBC_CONNECTION_URL = "jdbc:mysql://${var.host}:${var.port}/${var.database}"
    USERNAME            = var.username
    PASSWORD            = var.password
  }

  physical_connection_requirements {
    availability_zone      = data.aws_subnet.private_a.availability_zone
    security_group_id_list = [data.aws_security_group.db_sg.id]
    subnet_id              = data.aws_subnet.private_a.id
  }
}

resource "aws_glue_crawler" "s3_crawler" {
  name          = "${var.project}-analytics-db-crawler"
  database_name = aws_glue_catalog_database.analytics_database.name
  role          = aws_iam_role.glue_role.arn

  s3_target {
    path = "s3://${aws_s3_bucket.data_lake.bucket}/gold"
  }
```

# Terraform Configuration File (s3.tf)


1. Set up the S3 bucket
2. Configure the bucket and provide public access to it

# Terraform Language

```
resource "aws_s3_bucket" "data_lake" {
  bucket_prefix = "${var.project}-datalake-${data.aws_caller_identity.current.account_id}-"
}

resource "aws_s3_bucket_public_access_block" "data_lake" {
  bucket = aws_s3_bucket.data_lake.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_s3_bucket" "scripts" {
  bucket_prefix = "${var.project}-scripts-${data.aws_caller_identity.current.account_id}-"
}

resource "aws_s3_bucket_public_access_block" "scripts" {
  bucket = aws_s3_bucket.scripts.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_s3_object" "glue_job_script" {
  bucket = aws_s3_bucket.scripts.id
```

**Domain-Specific Language**

HCL (HashiCorp Configuration Language)

HashiCorp Terraform

# The HCL Syntax

```
resource "aws_vpc" "main" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "main"
  }
}
```

Create or update a VPC

Virtual private cloud (VPC)

```
resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

Create and Update an EC2 instance

Instance

# The HCL Syntax

```
resource "google_compute_instance" "default" {
  name         = "my-instance"
  machine_type = "n2-standard-2"
  zone         = "us-central1-a"

  tags = ["foo", "bar"]

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
      labels = {
        my_label = "value"
      }
    }
  }
```

Provision a GCP (Google Cloud Platform) compute instance

# Terraform Language

## HCL is a declarative language

You just have to declare what you want the infrastructure to look like

```
resource "aws_s3_bucket" "data_lake" {
  bucket_prefix = "${var.project}-datalake-${data.aws_caller_identity.current.account_id}-"
}

resource "aws_s3_bucket_public_access_block" "data_lake" {
  bucket = aws_s3_bucket.data_lake.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}
```

S3 Bucket

s3.tf

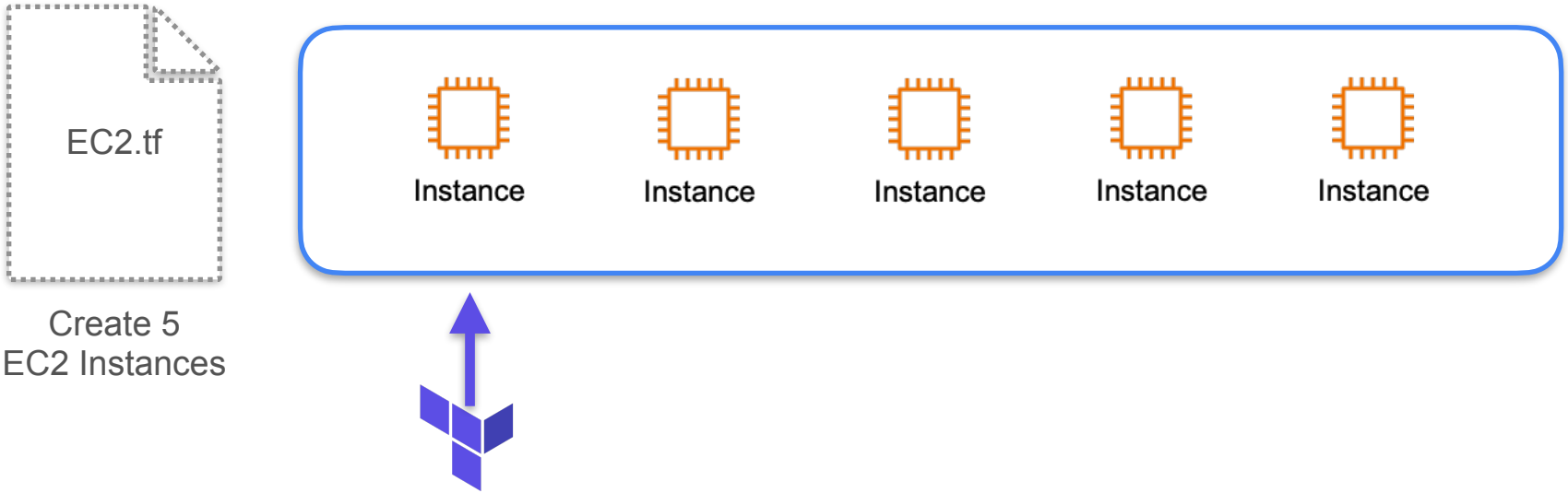
The desired end-state of the infrastructure:

*resources and their configurations*

**Terraform is Highly idempotent!**

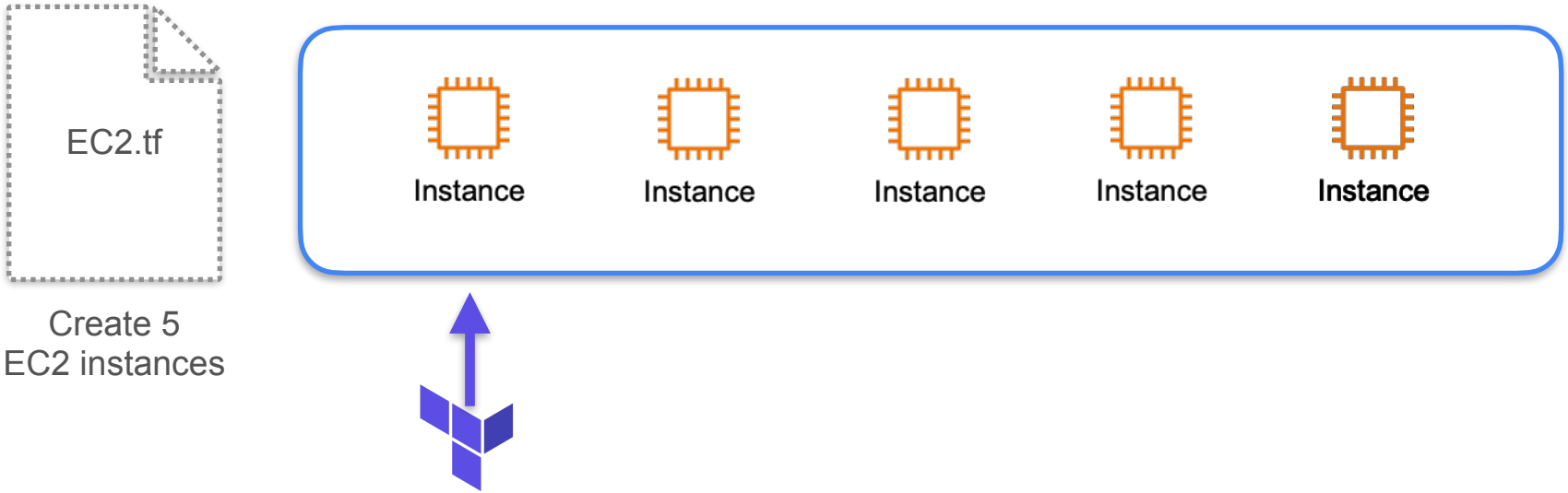
# Terraform Language

**HCL is a declarative language**



# Terraform Language

**HCL is a declarative language**



# Imperative/Procedural Language


**Imperative**: specify the exact sequence of configuration tasks.

DeepLearning.AI

# DataOps - Automation

## Terraform - Creating an EC2 instance

# Terraform

You write the configuration files to define your resources

Terraform prepares your workspace

1. Installs necessary files
2. Creates an execution plan

# Terraform

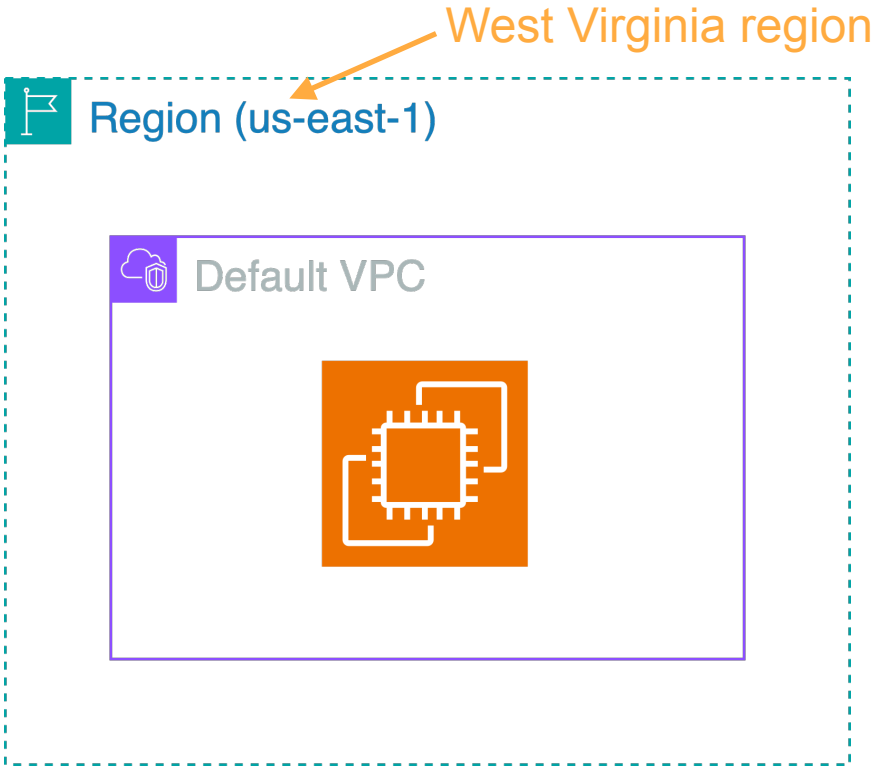
You write the configuration files to define your resources

Terraform prepares your workspace

You approve the execution plan

Terraform applies the proposed steps

# Terraform



# Terraform

Use any IDE of your choice

↓

Install Terraform in your environment

↓

Use AWS credentials to authenticate Terraform

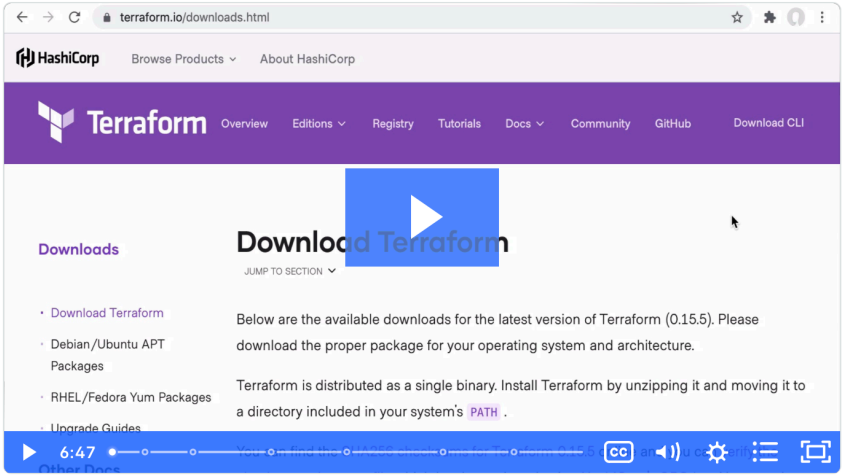
Developer / Terraform / Tutorials / AWS / Install

## Install Terraform

9min | Terraform ▷ Video Interactive

Show Terminal

Reference this often? Create an account to bookmark tutorials.




To use Terraform you will need to install it. HashiCorp distributes Terraform as a binary package. You can also install Terraform using popular package managers.

DeepLearning.AI

# DataOps - Automation

## Terraform - Defining variables and outputs

DeepLearning.AI

# DataOps - Automation

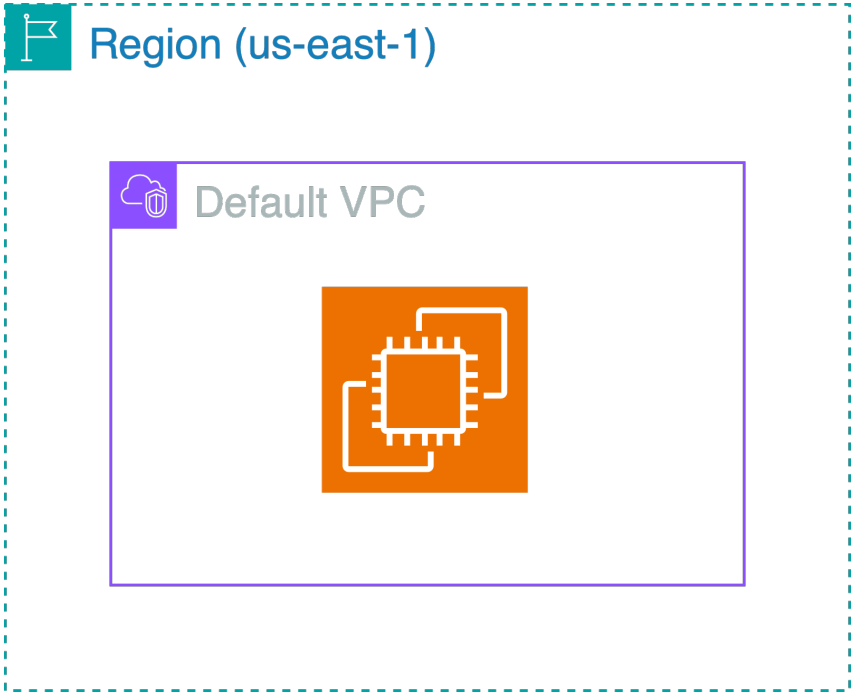
## Terraform - Defining data sources and modules

# Data Sources

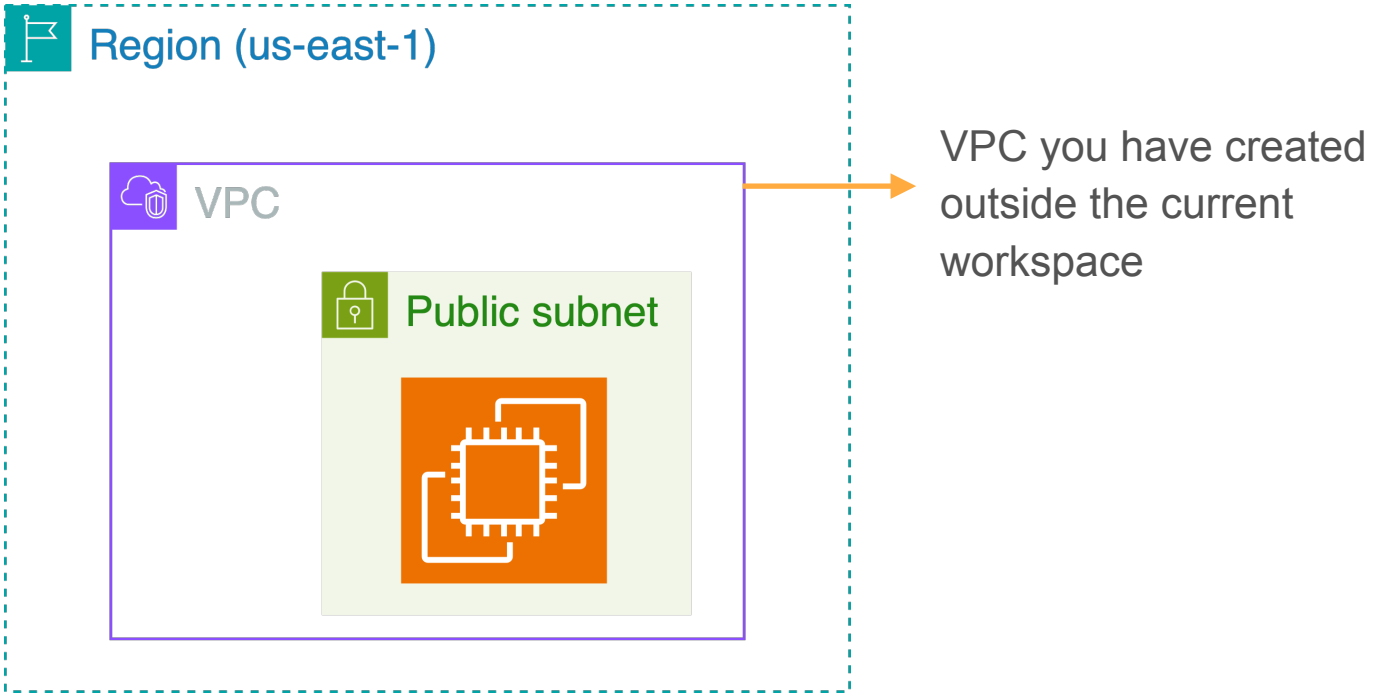
Data Sources

Data blocks to reference resources created outside Terraform or in another Terraform workspace.

# Data Sources



# Data Sources



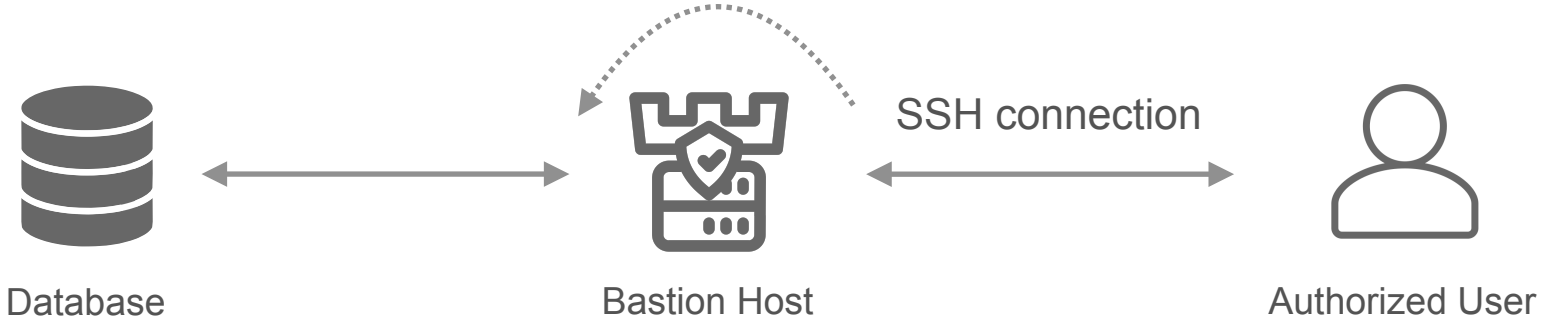
DeepLearning.AI

# Lab Walkthrough

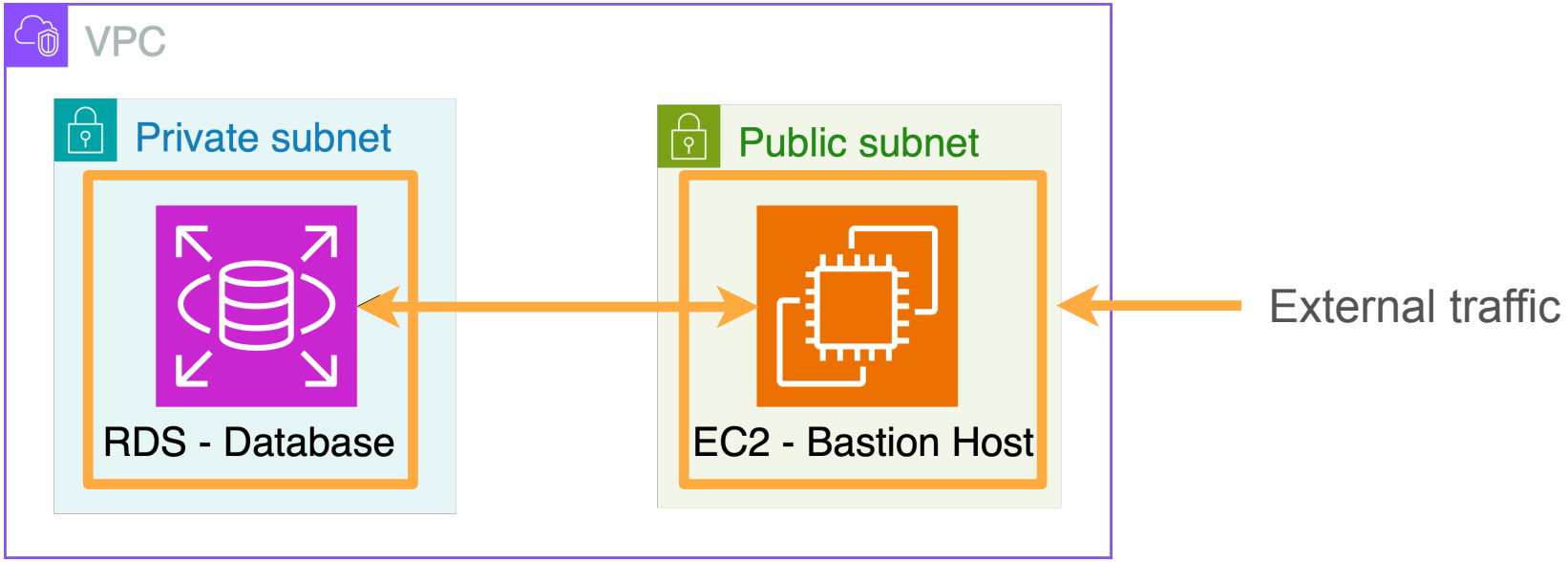
## Implementing DataOps Automation with Terraform

# Implementing DataOps with Terraform

- Architectural diagram of the resources
- Overview of the lab's steps

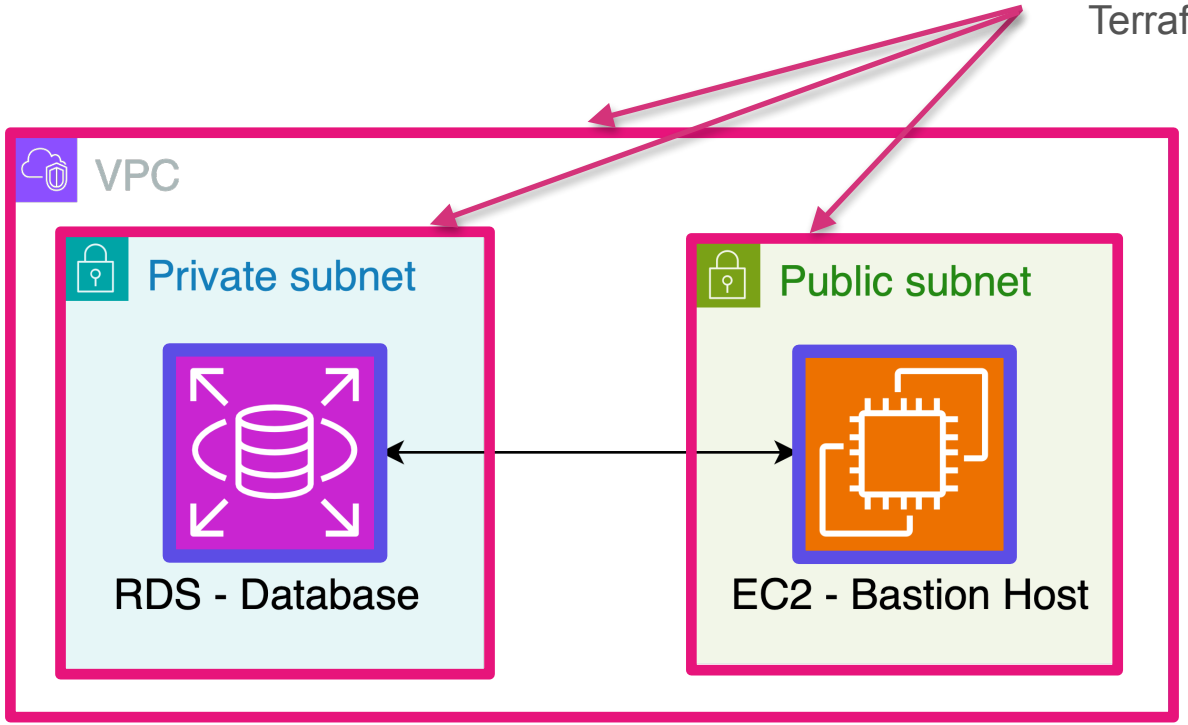


# Architectural Diagram



# Architectural Diagram

- Provided to you
- Defined as data blocks in Terraform configuration files

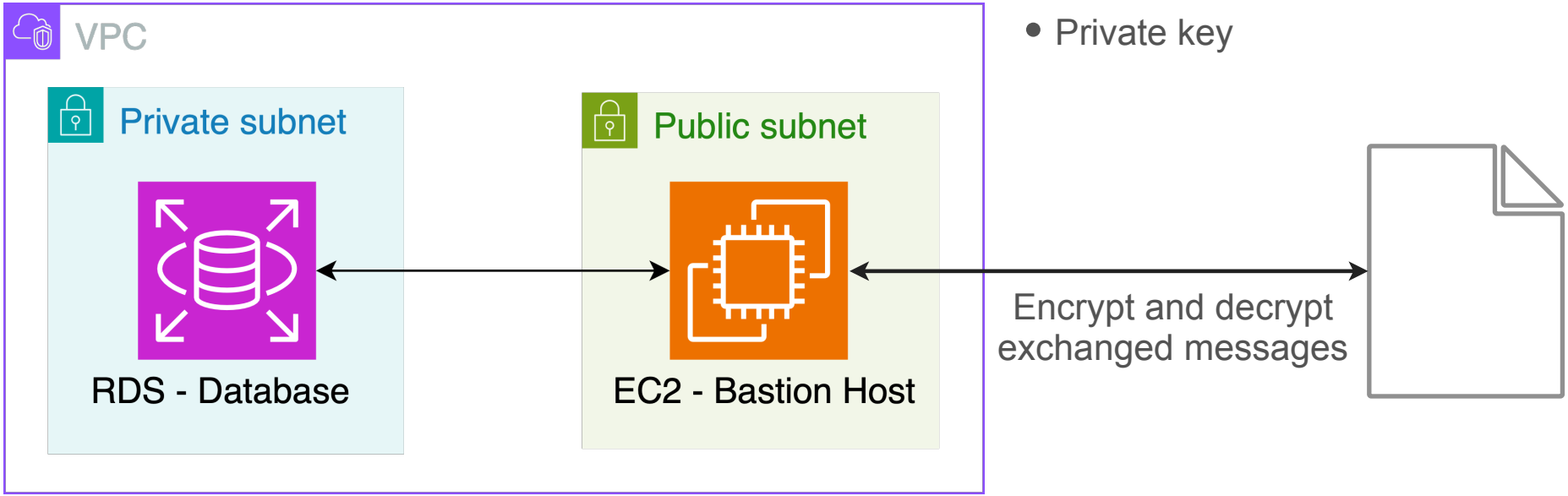



HashiCorp Terraform

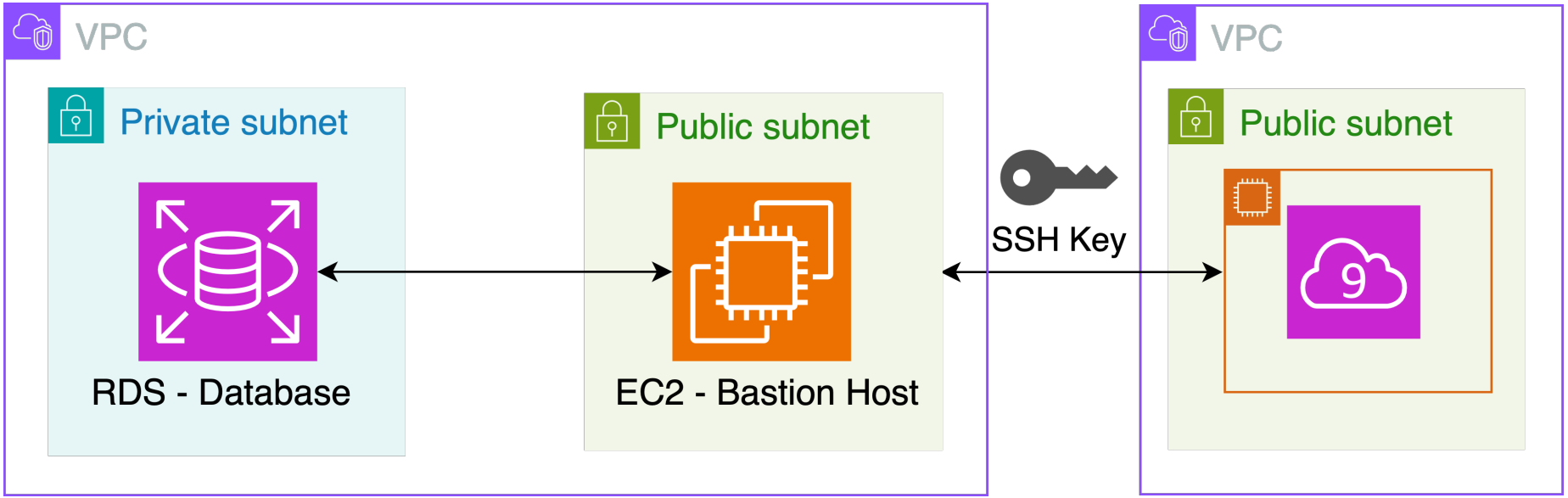
# Architectural Diagram

SSH Key Pair:

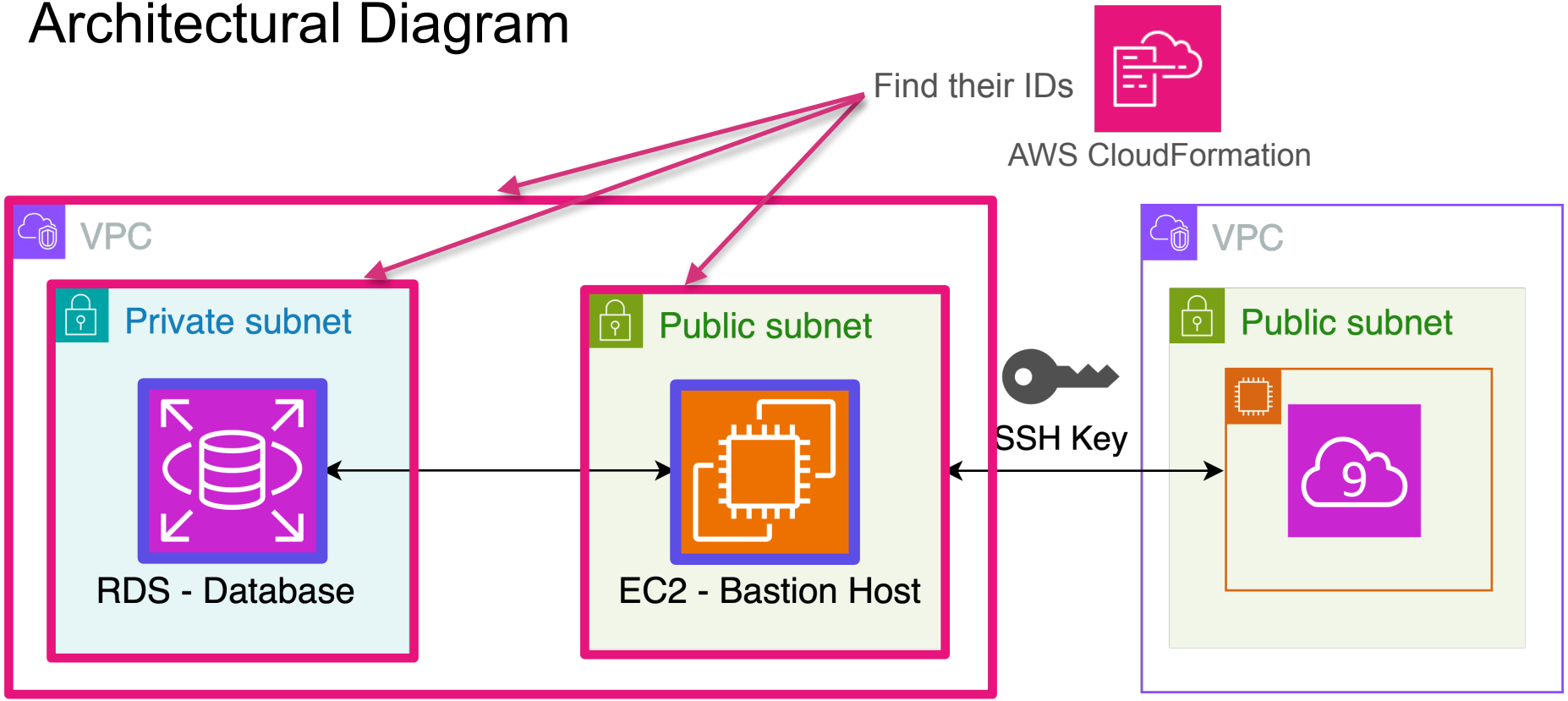
- Public key
- Private key



# Architectural Diagram

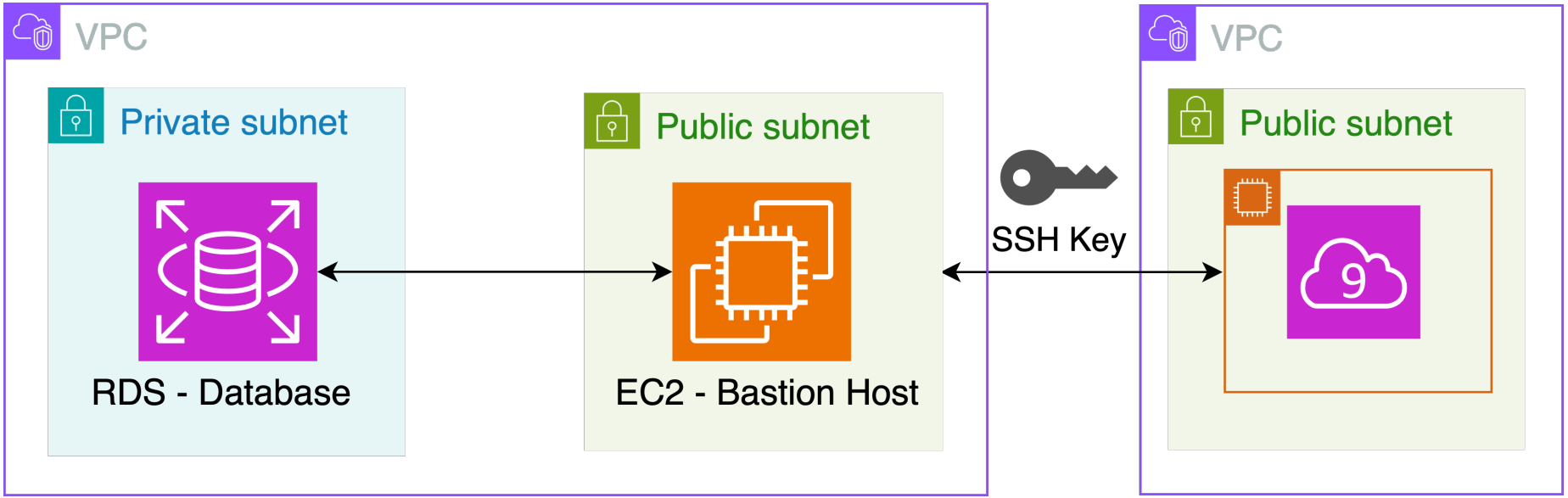


# Architectural Diagram



# Implementing DataOps with Terraform

Run terraform to create your resources

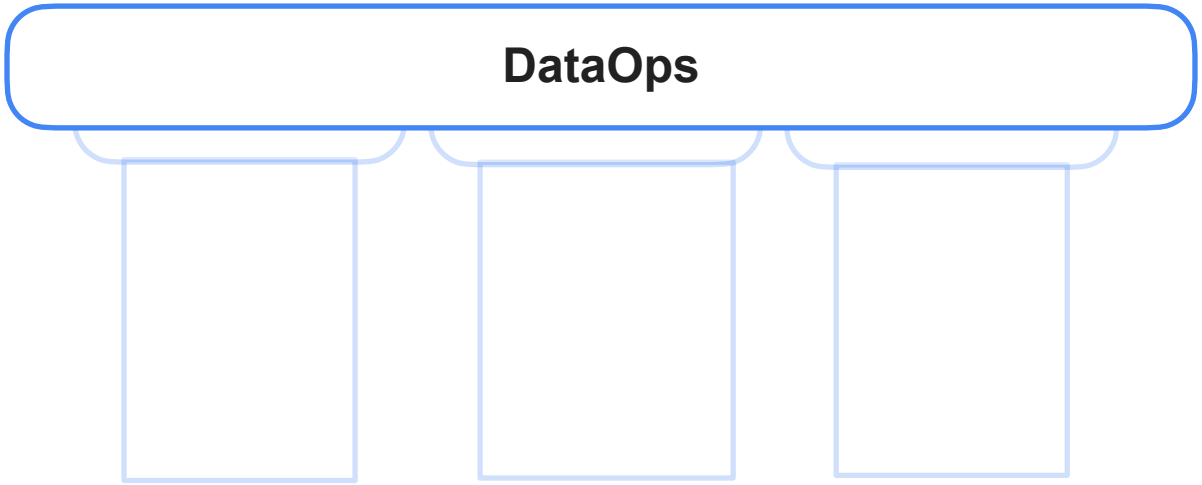


DeepLearning.AI

# DataOps - Observability & Monitoring

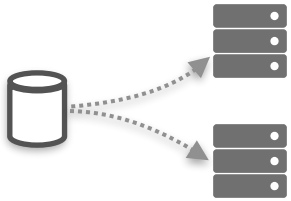
## Data Observability

# Observability and Monitoring



# DevOps Observability

Advent of the Cloud

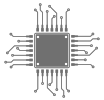


Move toward Distributed Systems

**Observability tools**

Gain visibility into systems' health

**Metrics**



CPU and RAM usage

Response time

- Quickly detect anomalies
- Identify problems
- Prevent downtime
- Ensure reliable software products

# Data Observability

**Monitor the health of data systems**

Observability tools

**Monitor the health and quality of data**

Health/quality of your data

# What is High Quality Data?

| High Quality Data | Low Quality Data |
| --- | --- |
| Accurate | Inaccurate |
| Complete | Incomplete |
| Discoverable | Hard to find |
| Available in a timely manner | Late |

Represents exactly what stakeholders expect:

- well-defined schema
- data definitions

# Quality of Data

Data Engineer

High quality data

Stakeholder

Value

Data Engineer

Low quality data

Stakeholder

**WORSE THAN**

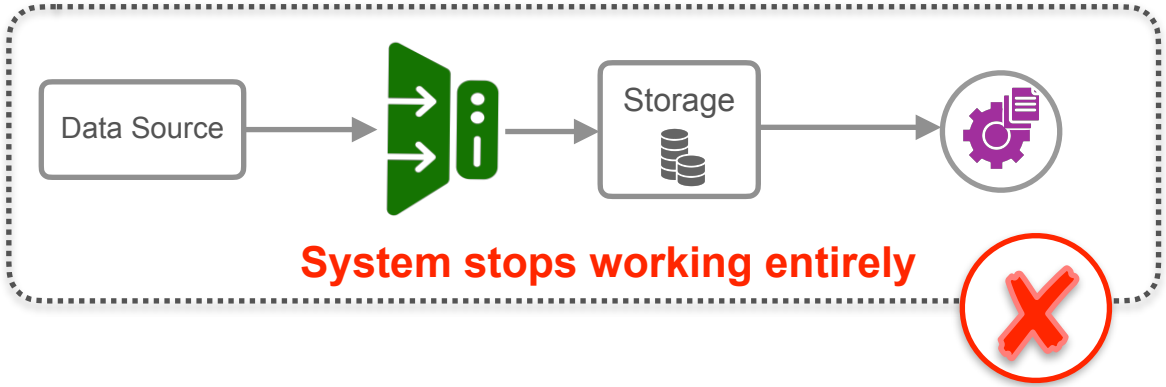
Data Engineer

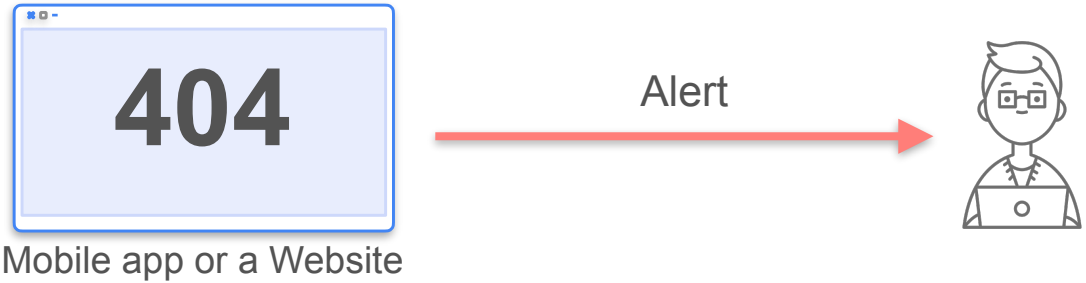
**NO DATA**

Stakeholder

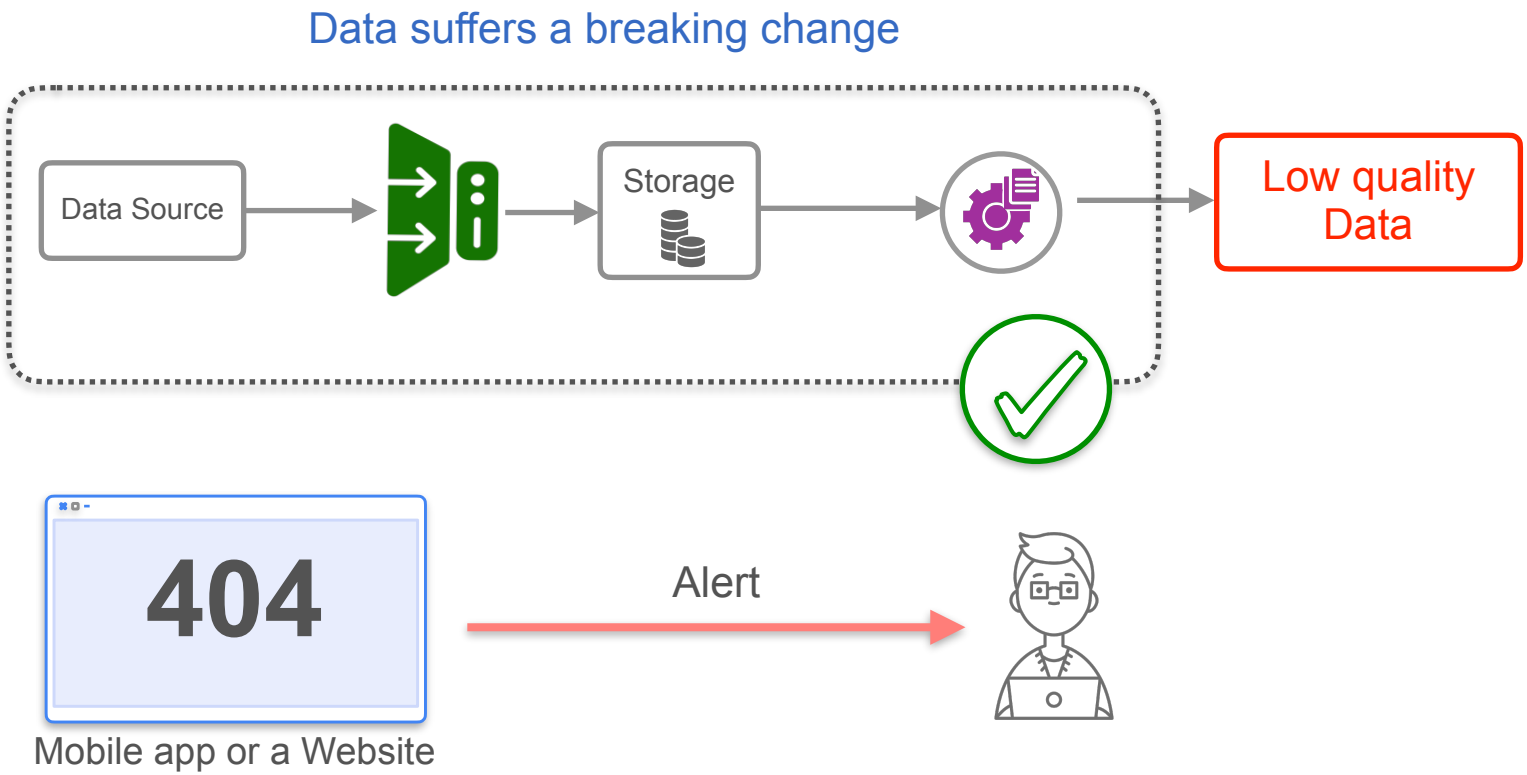
# Data Observability

System failure is your best case scenario



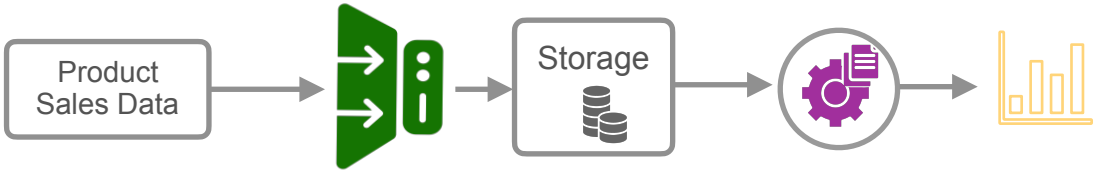
# Data Observability



# Scenario

U.S. based company

Sell products in Europe




| sale id | sale price (USD) |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 50 |

**Total**: 80

| sale id | sale price (EUR) |
|---|---|
| 1 | 9.15 |
| 2 | 18.31 |
| 3 | 45.77 |

**Total**:73.23

It looks like there was a sudden Drop in Revenue

DeepLearning.AI

# DataOps - Observability & Monitoring

**Barr Moses**

**Co-founder & CEO of Monte Carlo**

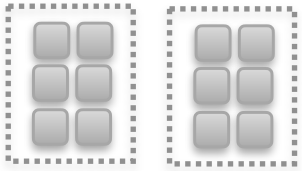
DeepLearning.AI

# DataOps - Observability & Monitoring

## Monitoring Data Quality

# Data Quality Metrics

- **Volume**: Total number of records ingested in each batch or over some time interval



- **Distribution**: Range of values in a particular column stays within some thresholds

| col 1 | col 2 | col 3 |
|---|---|---|
| 1 | 10 | 100 |
| 2 | 20 | 300 |
| 3 | 50 | 600 |

- **Null values**: Total number of null values in a table

| col 1 | col 2 | col 3 |
|---|---|---|
| 1 | 10 | 100 |
| 2 | null | null |
| null | 50 | 600 |

- **Freshness**: The *difference* in time between *now* and the *timestamp of the most recent* record in your data

- Identify and focus on the most important metrics
- Avoid creating confusion and “alert fatigue”

# Data Quality Metrics

Stakeholders

**What do stakeholders care about most for this use case?**

Stakeholder Requirement

Current data: no more than 24 hours old

What to monitor?

“Freshness” of the data: measure when the latest records were ingested

# Data Quality Metrics

## Stakeholders interested in product sales revenue

Sales

| sale id | product id | purchase amount |
|---|---|---|
| 1 | 2 | 10 |
| 2 | 3 | 20 |
| 3 | 4 | null |

Product

| product id | product name | SKU |
|---|---|---|
| 1 | product 1 | yk3 |
| 2 | product 2 | yk3 |
| 3 | product 3 | sj6 |

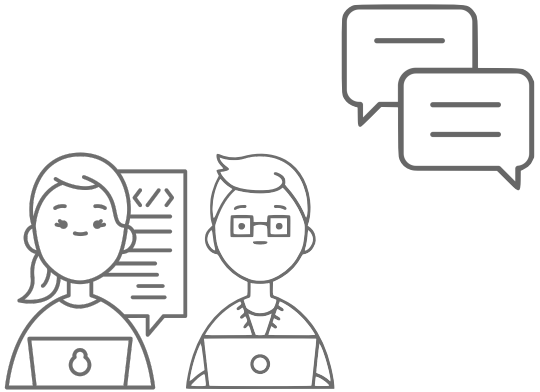
### What to monitor?

- Verify that all sales records were successfully ingested
- No null values in sales record

### What may be less important?

- Product SKU numbers match product descriptions
- Each customer's postal code was recorded correctly

# Testing Ingested Data



Source Owners

Data Engineer

- Build in checks or tests to verify that the schema and data types stay consistent.
- Identify problems as early as possible before they propagate further down your pipelines.

DeepLearning.AI

# DataOps - Observability & Monitoring

**Abe Gong**

**Co-founder & CEO of Great Expectations**

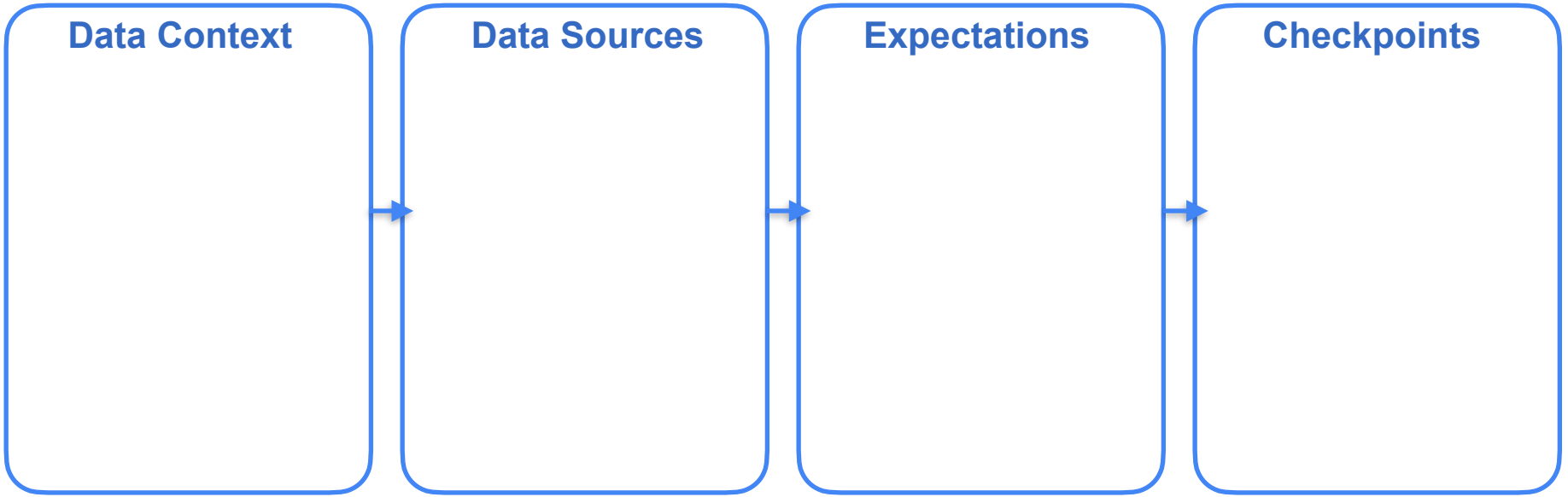
DeepLearning.AI

# DataOps - Observability & Monitoring

## Great Expectations - Core Components

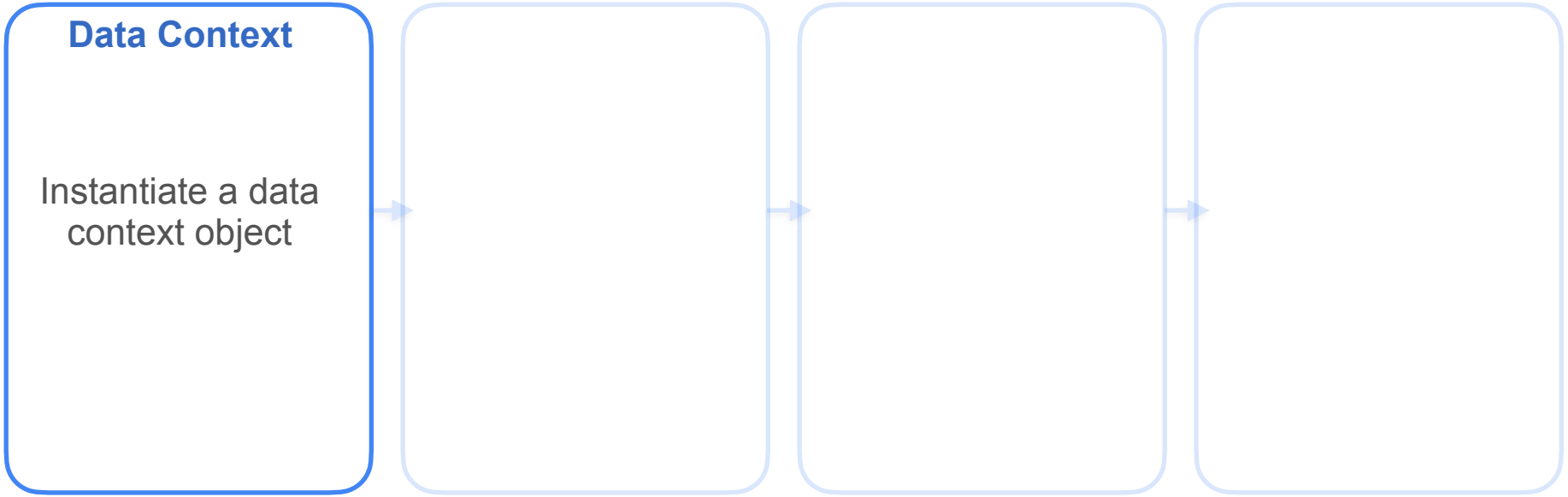
great expectations

1. Specify the data
2. Define your expectations
3. Validate your data against the expectations



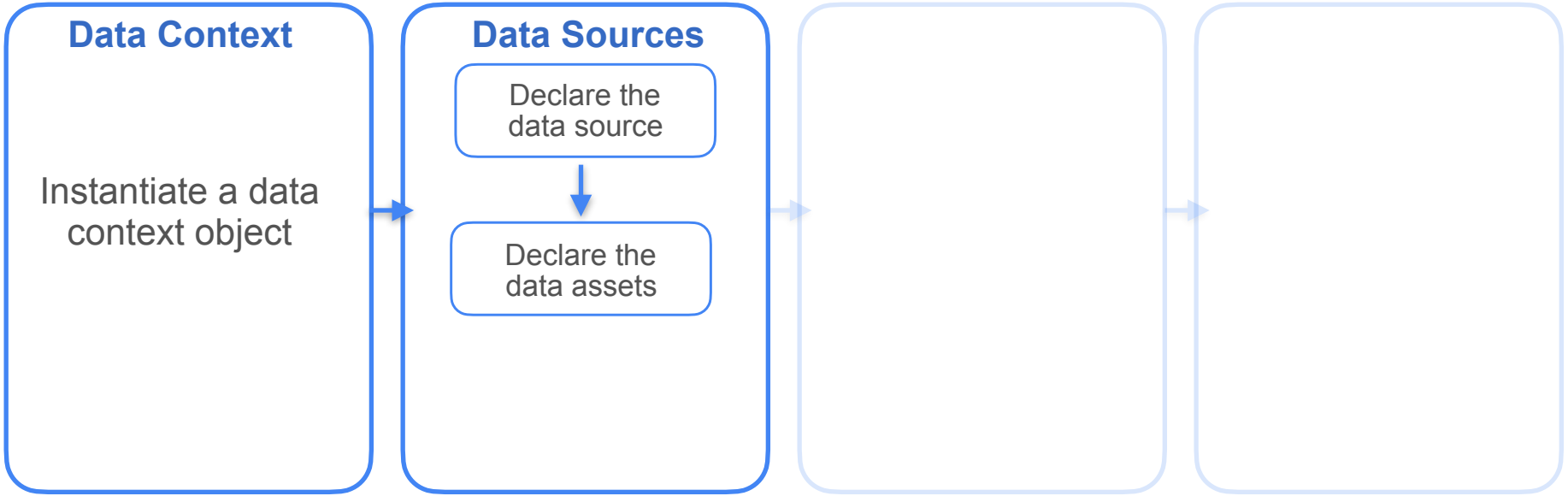
# Great Expectations Workflow

- **Data Context**: entry point for the Great Expectations API
- **Great Expectations API**: classes and methods that allow you to connect data sources, create expectations and validate your data



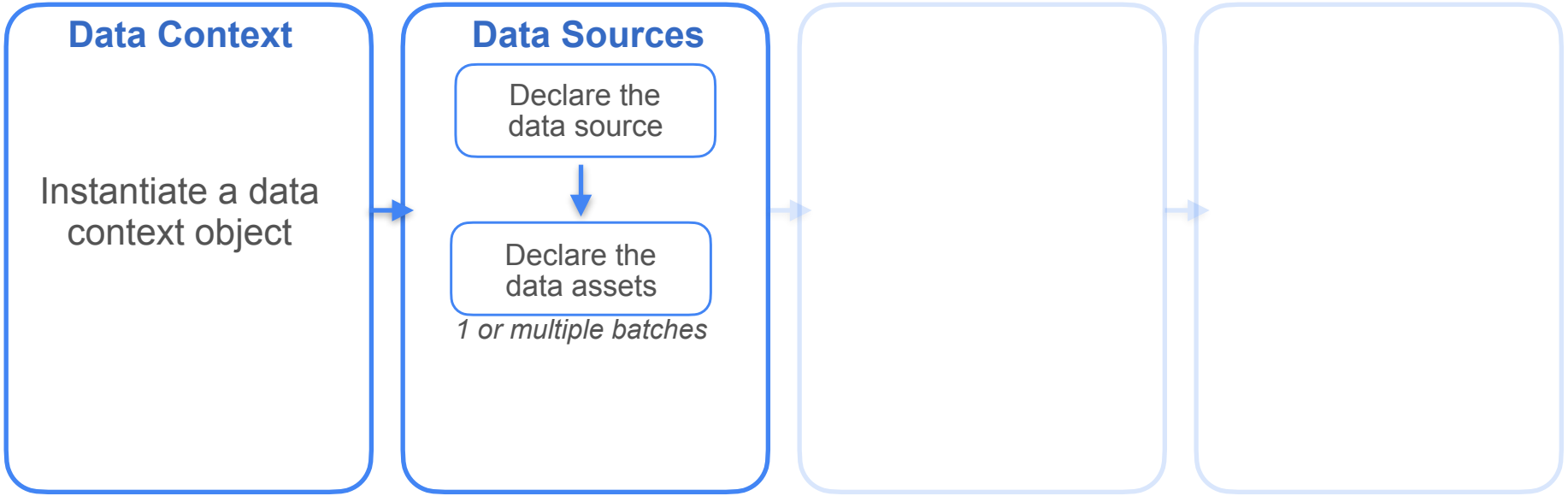
# Great Expectations Workflow

- **Data source:** SQL Database, a local file system, an S3 bucket, a Pandas DataFrame
- **Data asset**: collection of records within a data source
  - Table in a SQL database
  - File in a file system
  - Join query asset
  - Collections of files matching a pattern



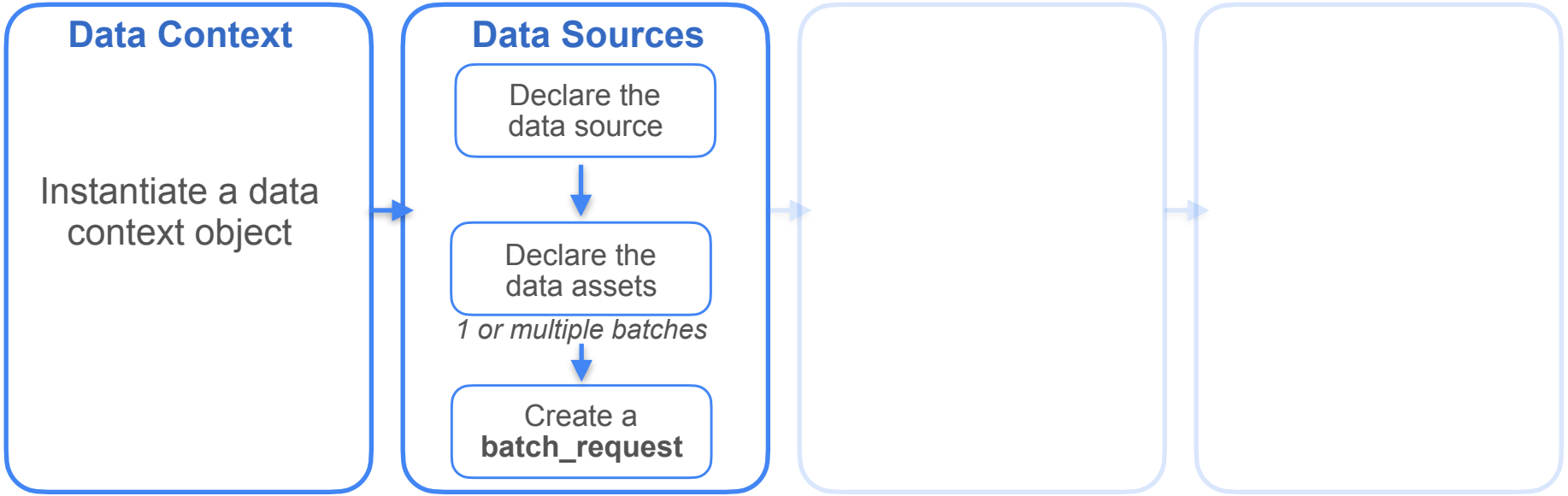
# Great Expectations Workflow

- **Data source:** SQL Database, a local file system, an S3 bucket, a Pandas DataFrame
- **Data asset**: collection of records within a data source
- **Batches**: partitions from your data asset
  - partitions by date
  - partitions by column values



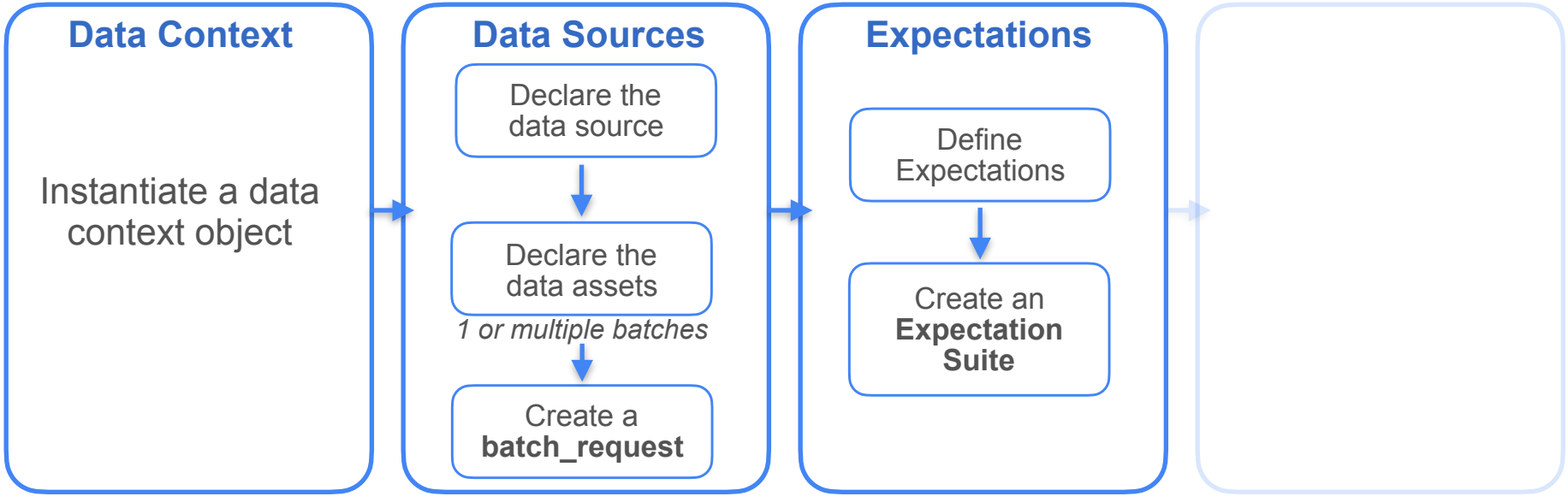
# Great Expectations Workflow

- **Data source:** SQL Database, a local file system, an S3 bucket, a Pandas DataFrame
- **Data asset**: collection of records within a data source
- **Batches**: partitions from your data asset
- **Batch_request**: primary way to retrieve data from the data asset



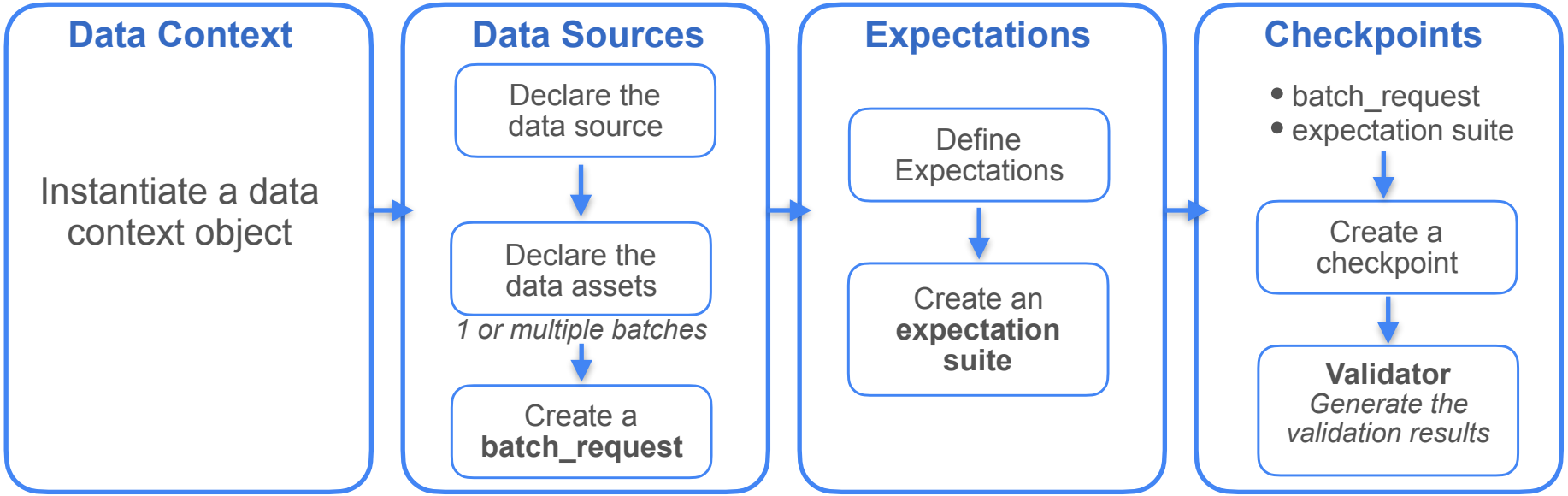
# Great Expectations Workflow

- **Expectation:** statement that you can use to verify if your data meets a certain condition
  - expect_column_min_to_be_between
  - expect_column_values_to_be_unique
  - expect_column_values_to_be_null



# Great Expectations Workflow

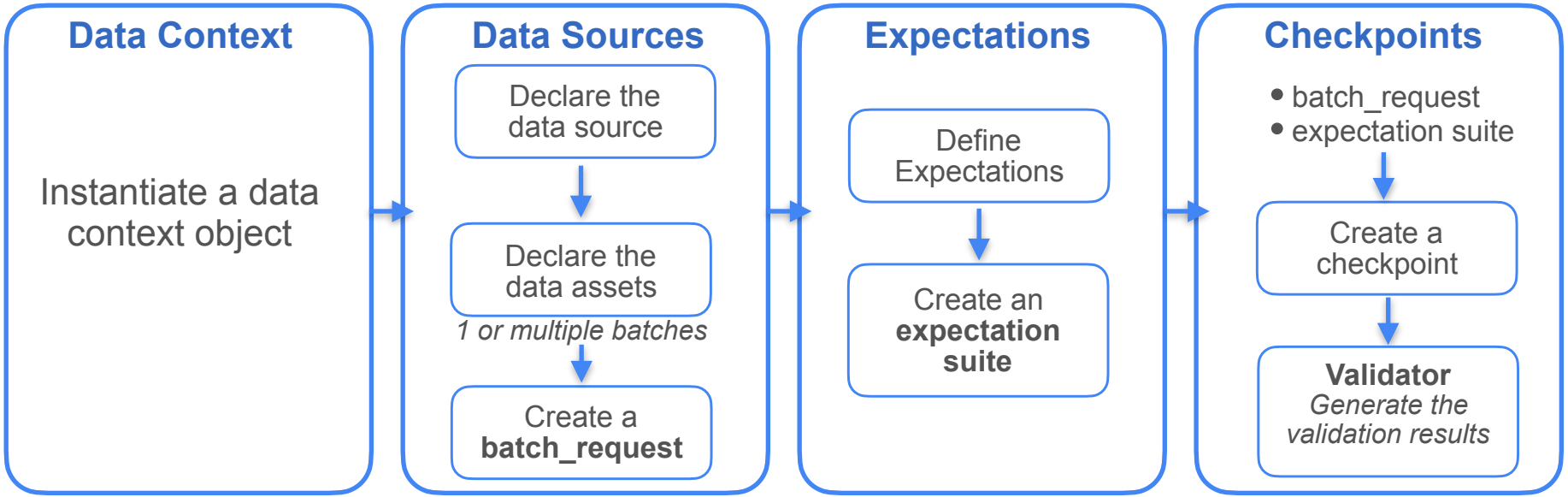
- **Validator:** expects a batch_request and its corresponding expectation suite
  - Manual interaction
  - Or automating the validation process using a checkpoint object



# Great Expectations Workflow

Metadata is saved in backend stores:

- Expectation Store
- Validation Store
- Checkpoint Store
- Data docs



DeepLearning.AI

# DataOps - Observability & Monitoring

## Great Expectations - Workflow Example

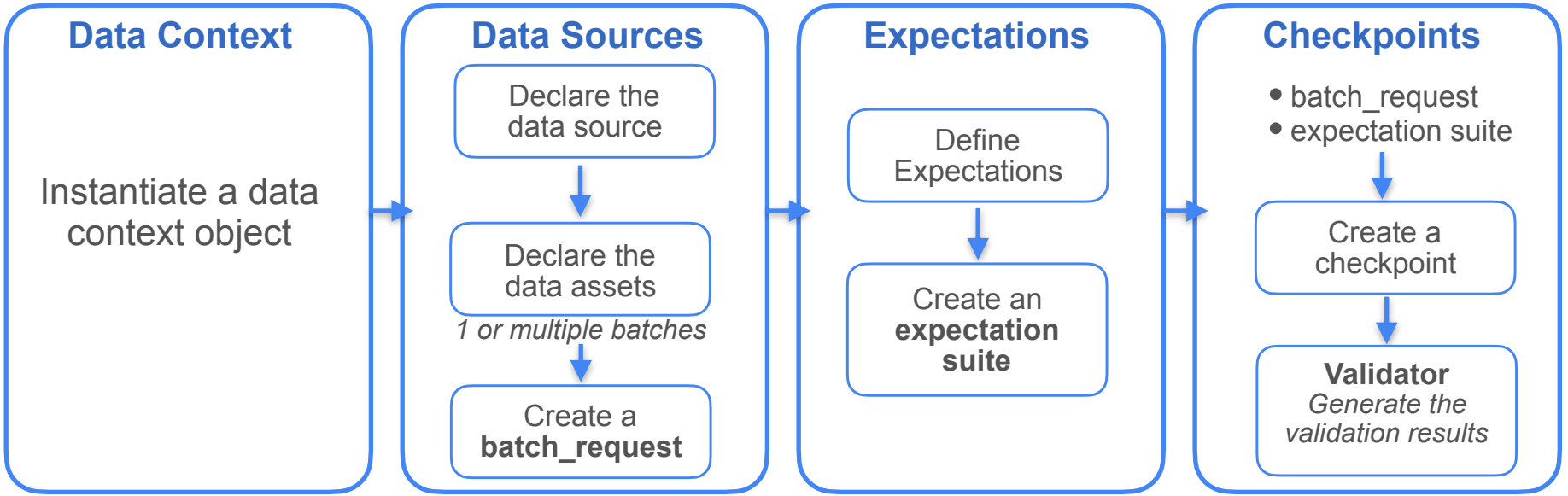
# Workflow Example

**DVD rental Database**

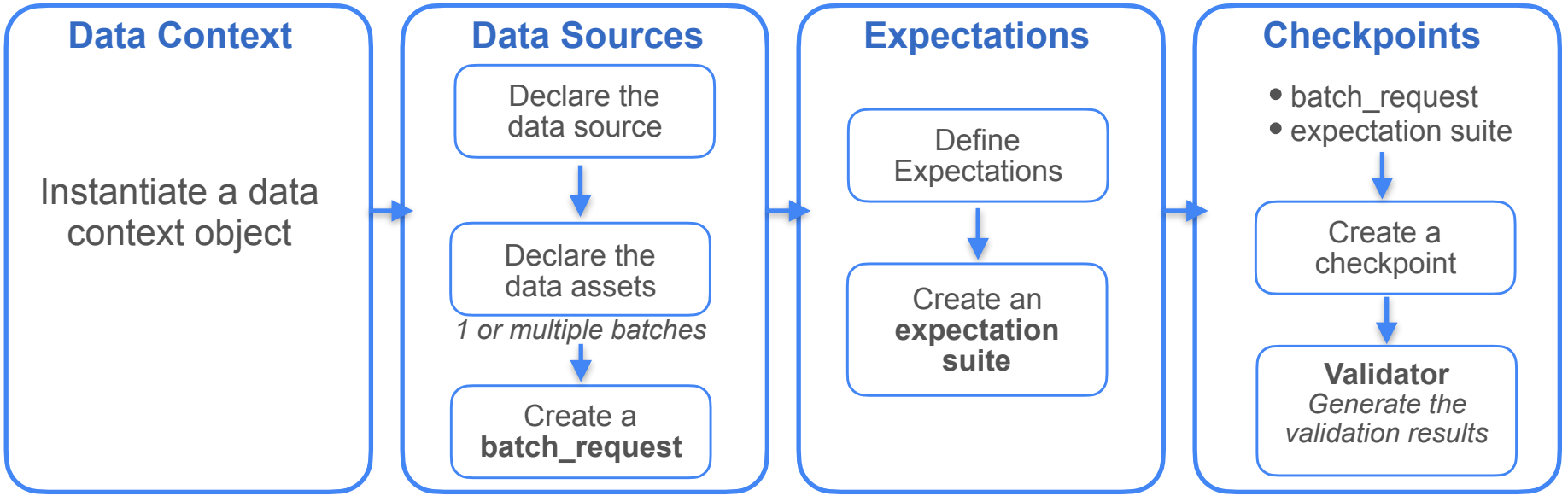
Local postgresSQL database

| payment |
| --- |
| payment_id |
| customer_id |
| staff_id |
| rental_id |
| amount |
| payment_date |

- payment_id contains unique IDs
- customer_id does not contain null values
- all the values in the amount column are non-negative



# Workflow Example



DeepLearning.AI

# DataOps - Observability & Monitoring

**Chad Sanderson**
**CEO of Gable.ai**

DeepLearning.AI

# DataOps - Observability & Monitoring

## Amazon CloudWatch

# Amazon CloudWatch

**System Level Metrics:**

- CPU utilization
- Disk I/O
- Network traffic
- Memory usage

# Amazon CloudWatch

Made aware when metrics reflect issues

**CloudWatch Alarms**

## System Level Metrics:

Provide a general understanding of how your resources are performing

- CPU utilization
- Disk I/O
- Network traffic
- Memory usage

## Custom Metrics:

Allow you to monitor specific aspects of your application that are not covered by default system metrics

- Number of transactions processed
- Response time of an API endpoint
- Number of active users

- Define thresholds for these metrics
- Establish a baseline

  Measure performance of your system under different loads and conditions
- CloudWatch retains metrics data for up to 15 months

# Common Metrics for RDS

**CPU Utilization**

- High value: your RDS instance might be under heavy load
- Values over 80-90%: can lead to performance bottlenecks

**RAM Consumption**

- High RAM consumption: can slow down performance

**Disk Space**

- Value consistently above 85%: you may need to delete or archive data to free up some space

**Database Connections**

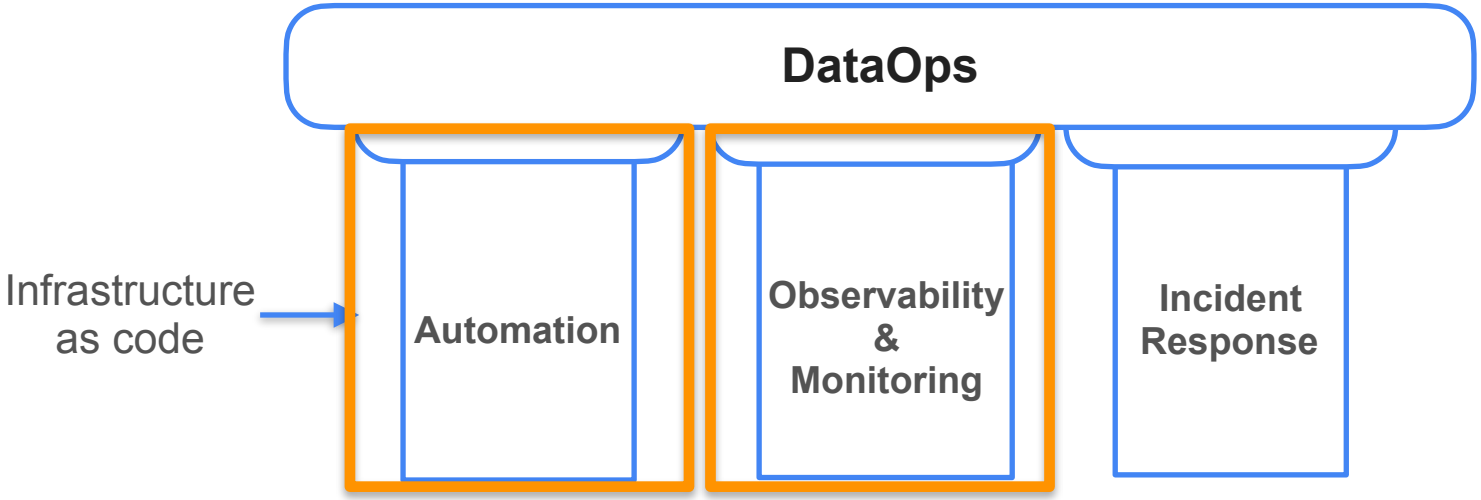
- The number of active connections to your database
- Number of connections approaching the maximum limit: can lead to connection errors and application failures

DeepLearning.AI

# DataOps

## Summary

# DataOps Summary




Ensure that you're delivering value for stakeholders.

# DataOps Summary

Chris Bergh

Barr Moses

Abe Gong

Chad Sanderson

# In This Week's Labs

HashiCorp Terraform

- A cloud agnostic tool for developing and maintaining your data infrastructure as a codebase
- Infrastructure as Code tools help make deployments consistent and repeatable

Amazon CloudWatch

great expectations