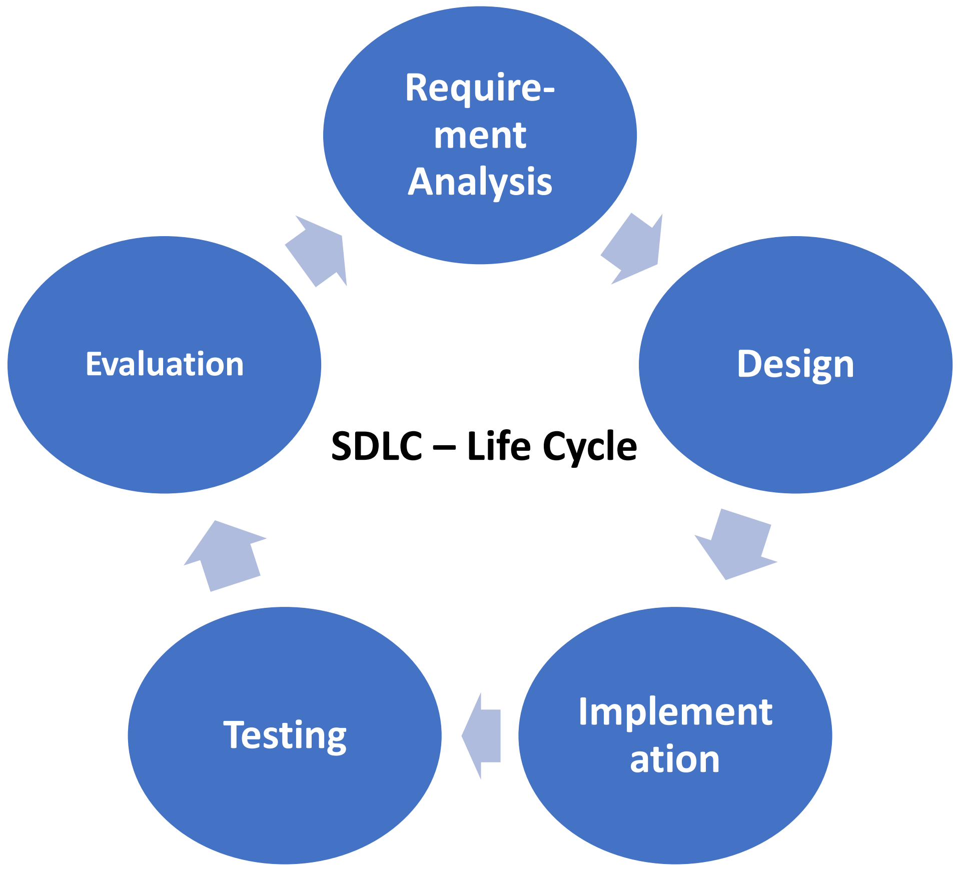
DevOps

What is DevOps?

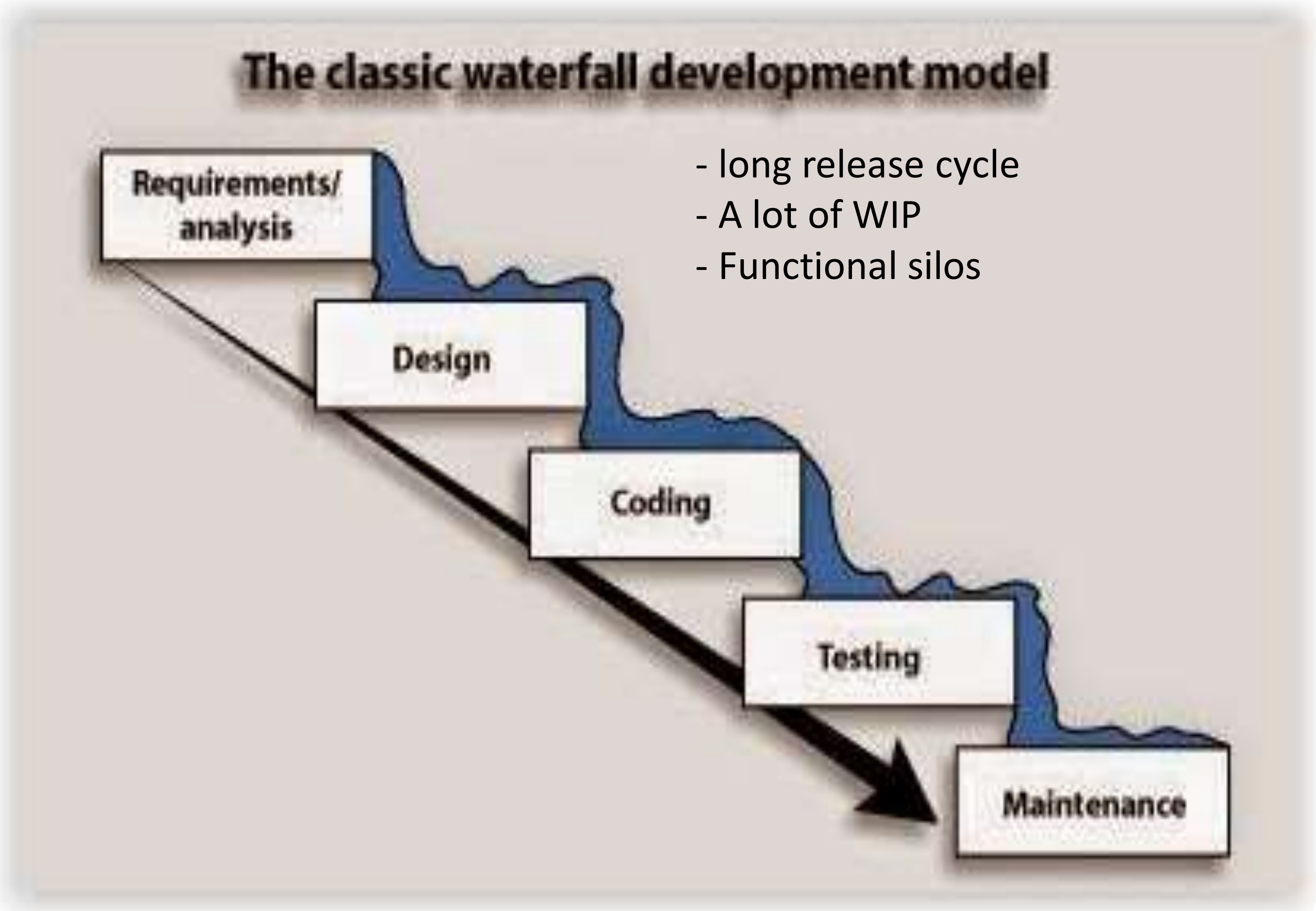
SDLC Model

- A systems development life cycle is composed of **several clearly defined and distinct work phases** which are used by systems engineers and systems developers to plan for, design, build, test, and deliver information systems



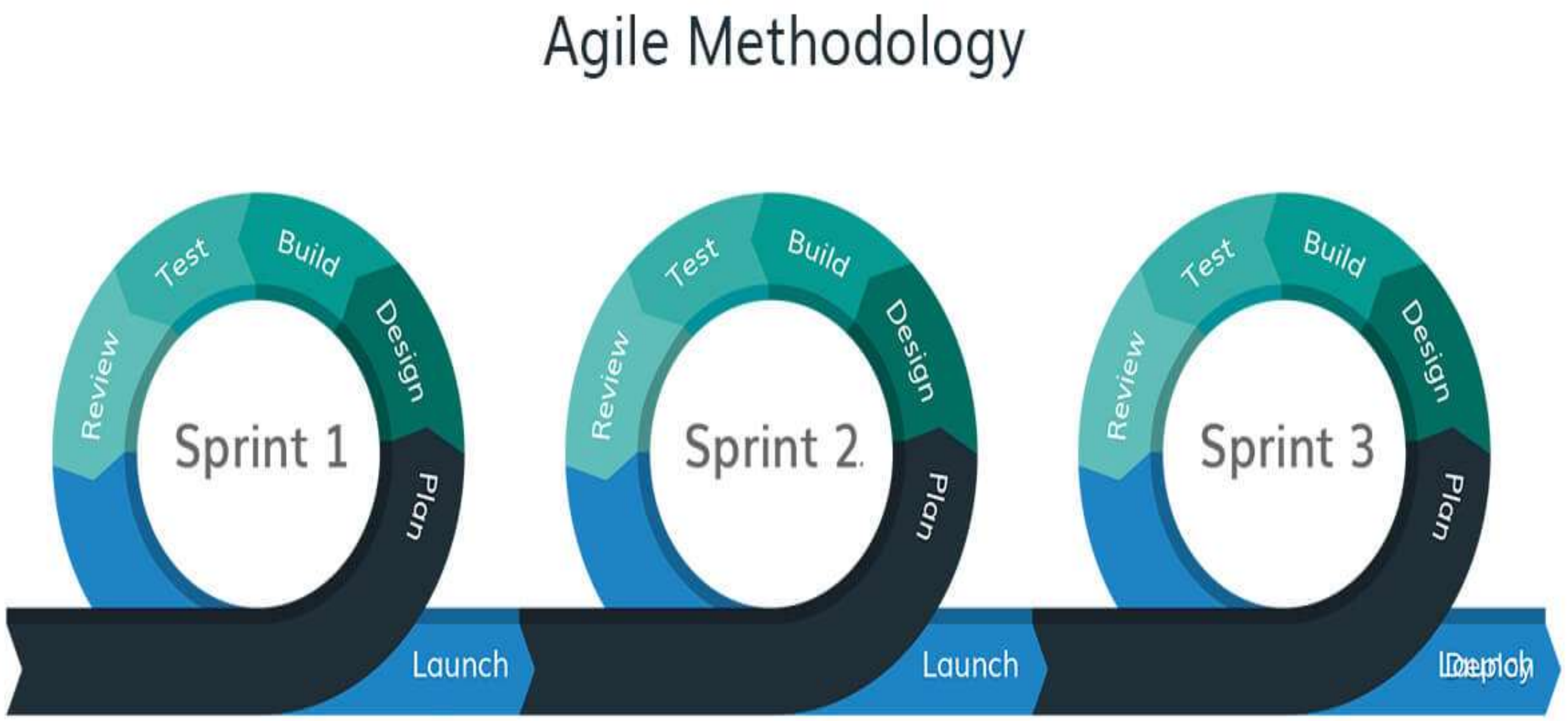
Waterfall Model

1. Determine the Requirements
2. Complete the design
3. Do the coding and testing (unit tests)
4. Perform other tests (functional tests, non-functional tests, Performance testing, bug fixes etc.)
5. At last deploy and maintain



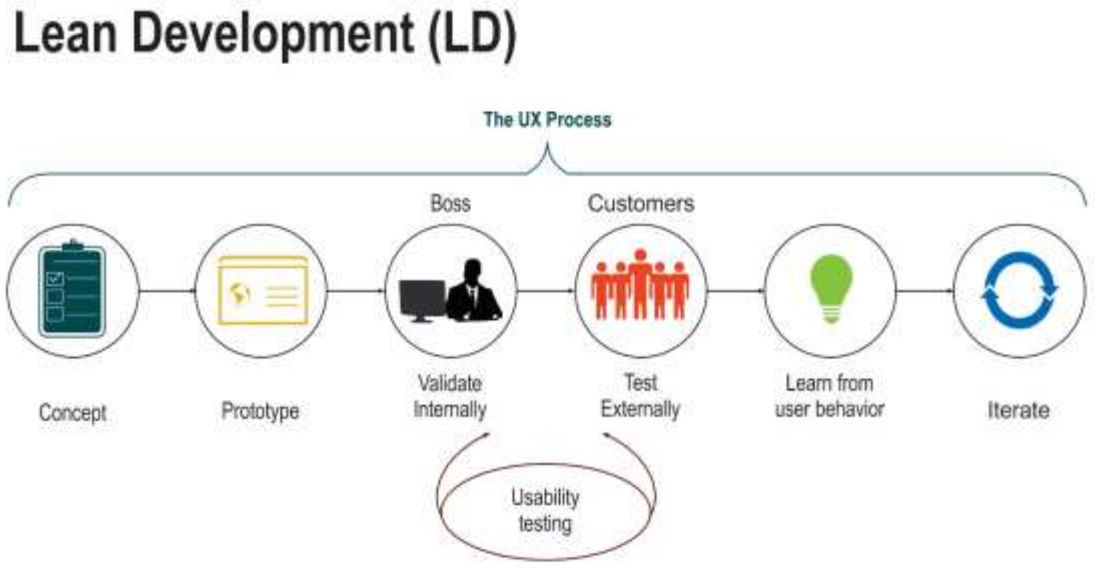
- long release cycle
- A lot of WIP
- Functional silos

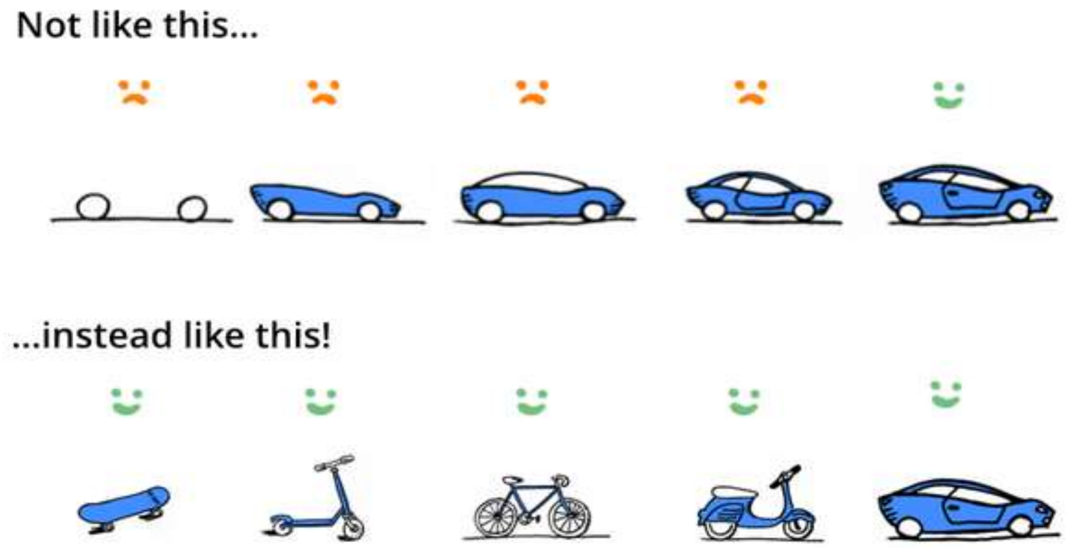
Agile



- Shorter release cycle
- Small batch sizes (MVP)
- Cross-functional teams
- Incredibly agile

Lean Development

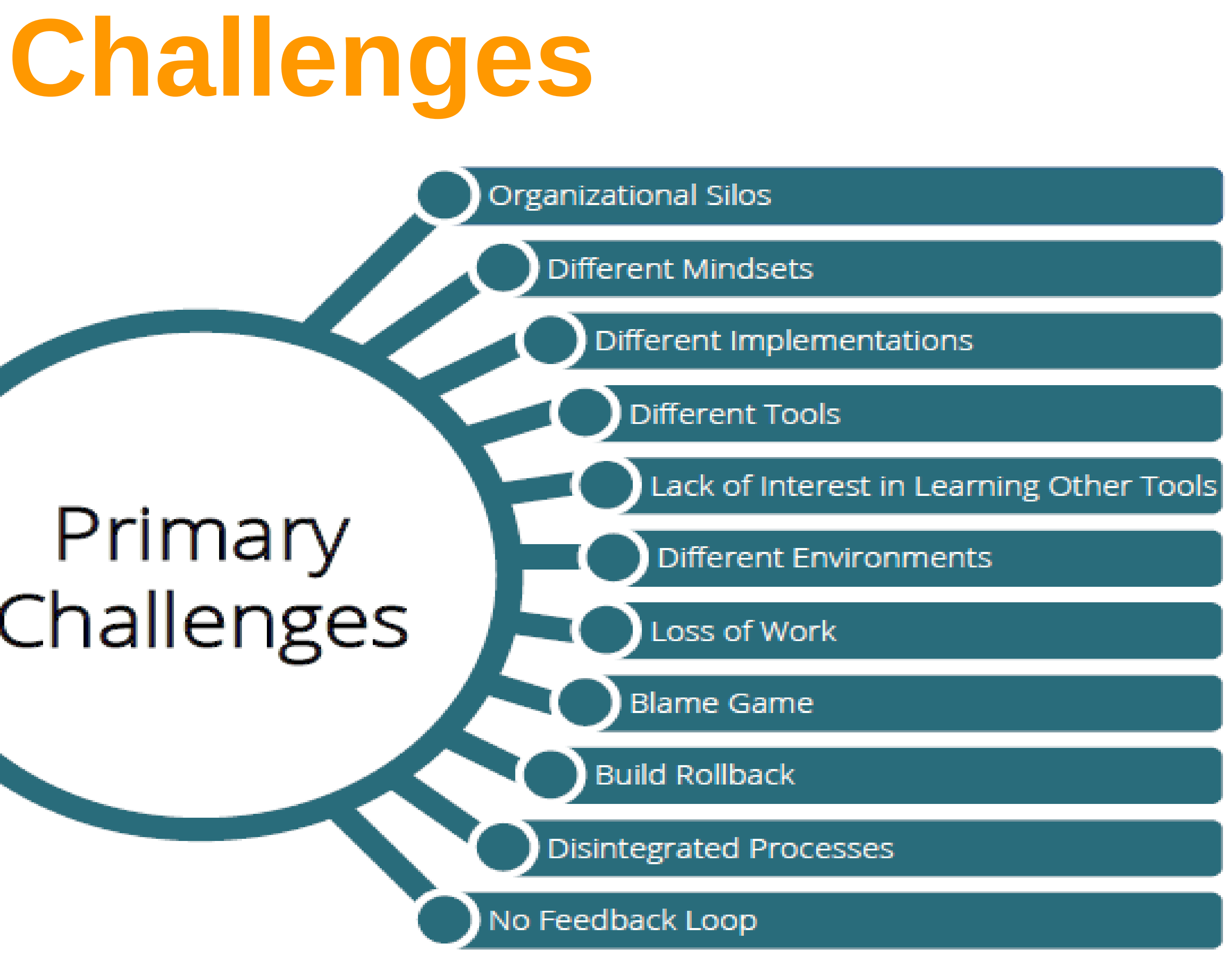




- Suddenly ops was the bottleneck (more release less people), again WIP is more!

Challenges

Some of the challenges with the traditional teams of Development and Operations are:



A Typical Case Study

- **Development Team:**

- Monday Morning, the writing of code done, unit tests completed, code delivered to the Integration teams to get the code included in CI builds.
- To get the services tested, a ticket is opened for QA teams

- **Build/Release/Testing/Integration Team:**

- Tuesday Morning, ticket accepted, a tester put an email to the developer asking deployment instructions. There is not automated deployments, developer updated to the tester, lets come online and we will deploy the services to the QA environment together.
- Call started, developer identified the “test environment” is not compatible.
- Tuesday afternoon, a ticket raised in Ops Team with new specifications.

- **Ops Team:**

- Wednesday morning, ticket accepted, specifications checked , a new port open request was identified.
- Ticket raised for Security team, ticket accepted, change approved, port opened, email received by the Ops team the work is done.

A Typical Case Study

- **Ops Team:**

- Identified the provisioning requirements again and started work on building the environment.

- **Build/Release/Testing/Integration Team:**

- Thursday Morning, updates received - the environment is ready. Developer and Tester again on call to deploy new services. Services deployed; tester is running test scripts. Next phase is to run regression test cases. Again a new ticket is raised for new test data with production teams and day ends.

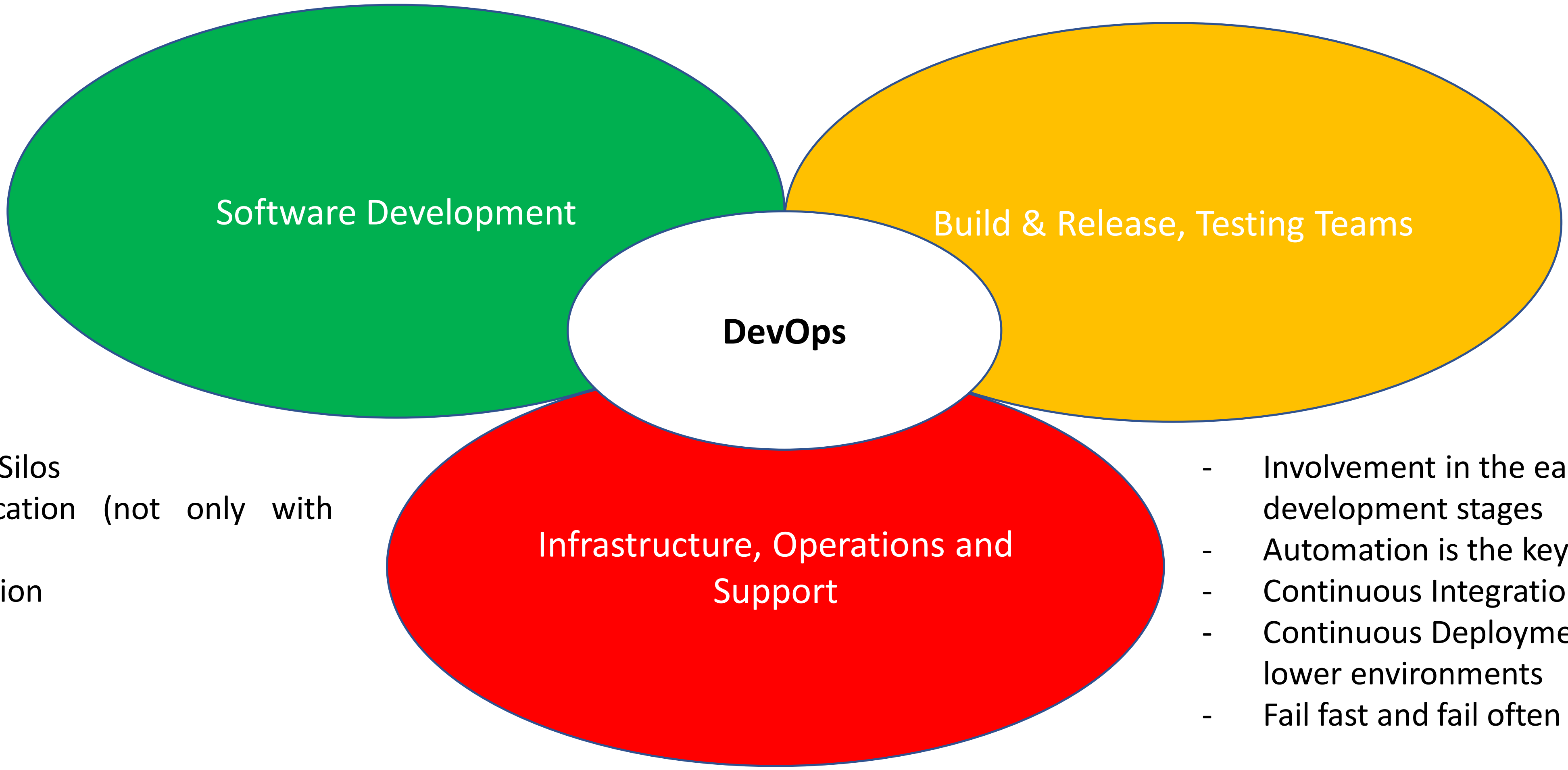
- **Ops Team:**

- Its Friday and the work is not on full swing, ticket accepted but not worked as production team has to complete rest of the works. Somehow the test data is gathered by Friday Evening.

- **Build/Release/Testing/Integration Team:**

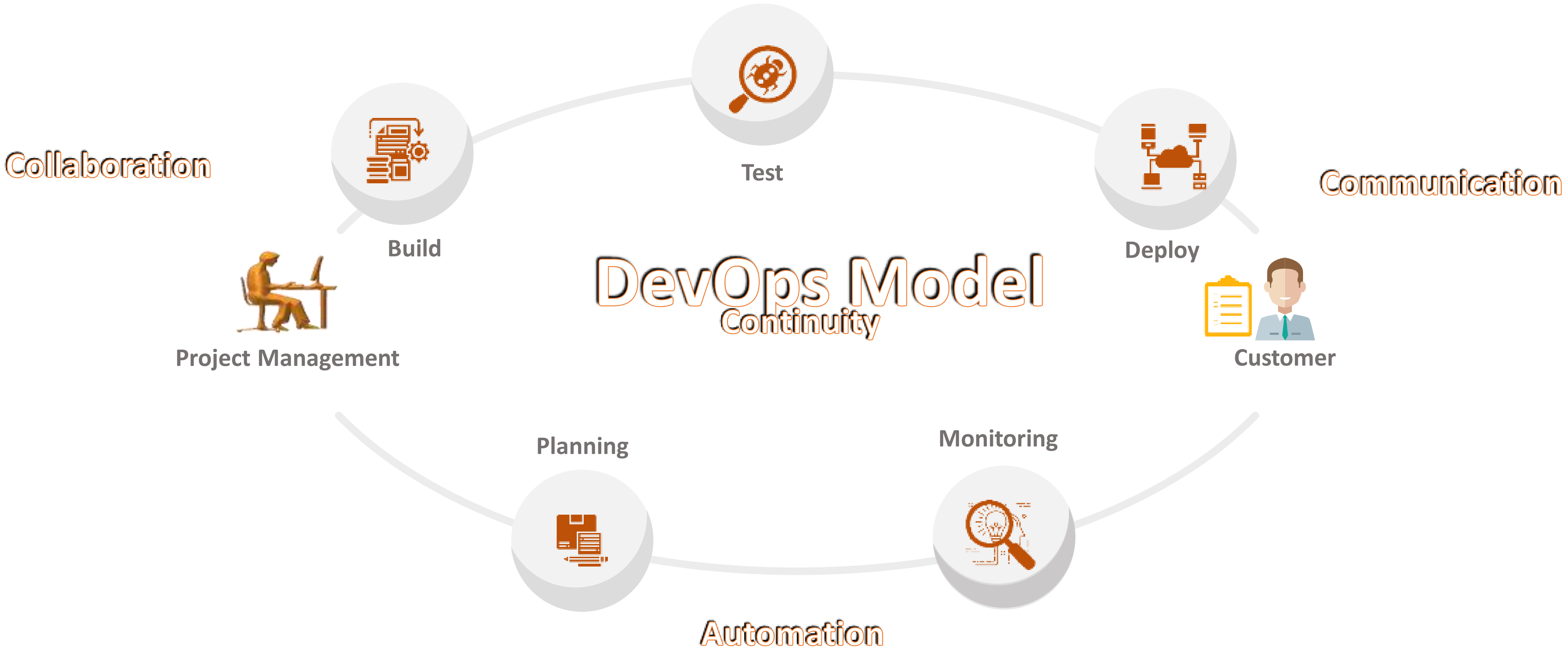
- Monday morning, tester gets the data, regression tests run, a defect found, and ticket returned to the development team.

DevOps


Software Development
Build & Release, Testing Teams
DevOps
Infrastructure, Operations and Support

- Break the Silos
- Communication (not only with emails)
- Collaboration
- Trust

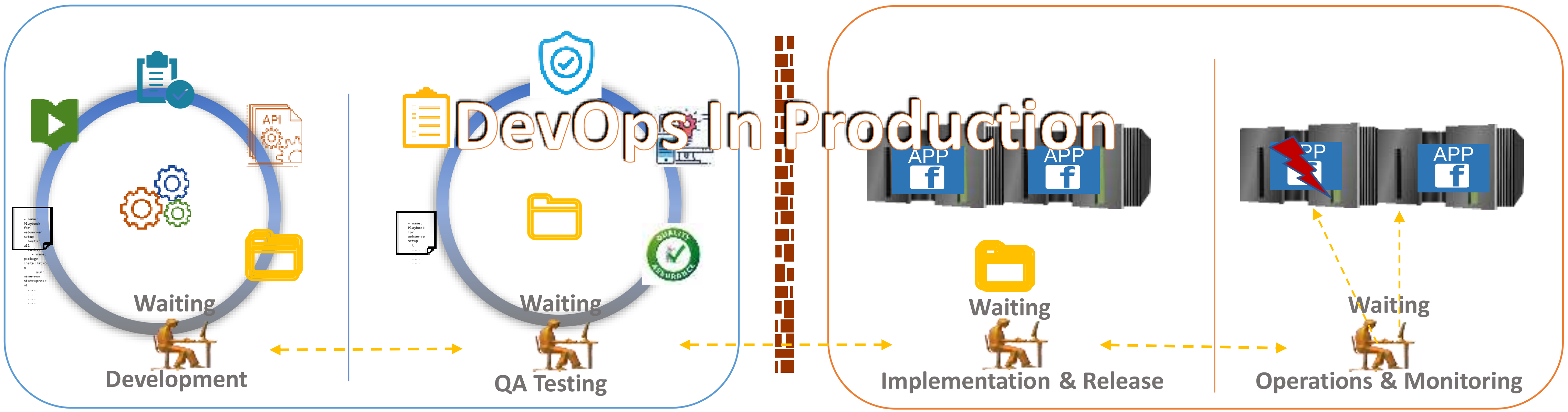
- Involvement in the early development stages
- Automation is the key
- Continuous Integration
- Continuous Deployments in the lower environments
- Fail fast and fail often


DevOps Model
Continuity
Collaboration
Communication
Automation
Test
Build
Deploy
Planning
Monitoring
Project Management
Customer

Continuous Feedback Continuous Improvement Continuous Planning

ContinuousContinuous Delivery Continuous Deployment Continuous Monitoring

DevOps In Production



Development Operations

DevOps Essence

Efficiency - Faster time to market

Predictability - Lower failure rate of new releases

Reproducibility – Version everything

Maintainability - Faster time to recovery in the event of a new release crashing or otherwise disabling the current system

How to Build DevOps Organization Culture

Retention is as important as recruitment

Establish Cross-functional team structure

Small teams are better

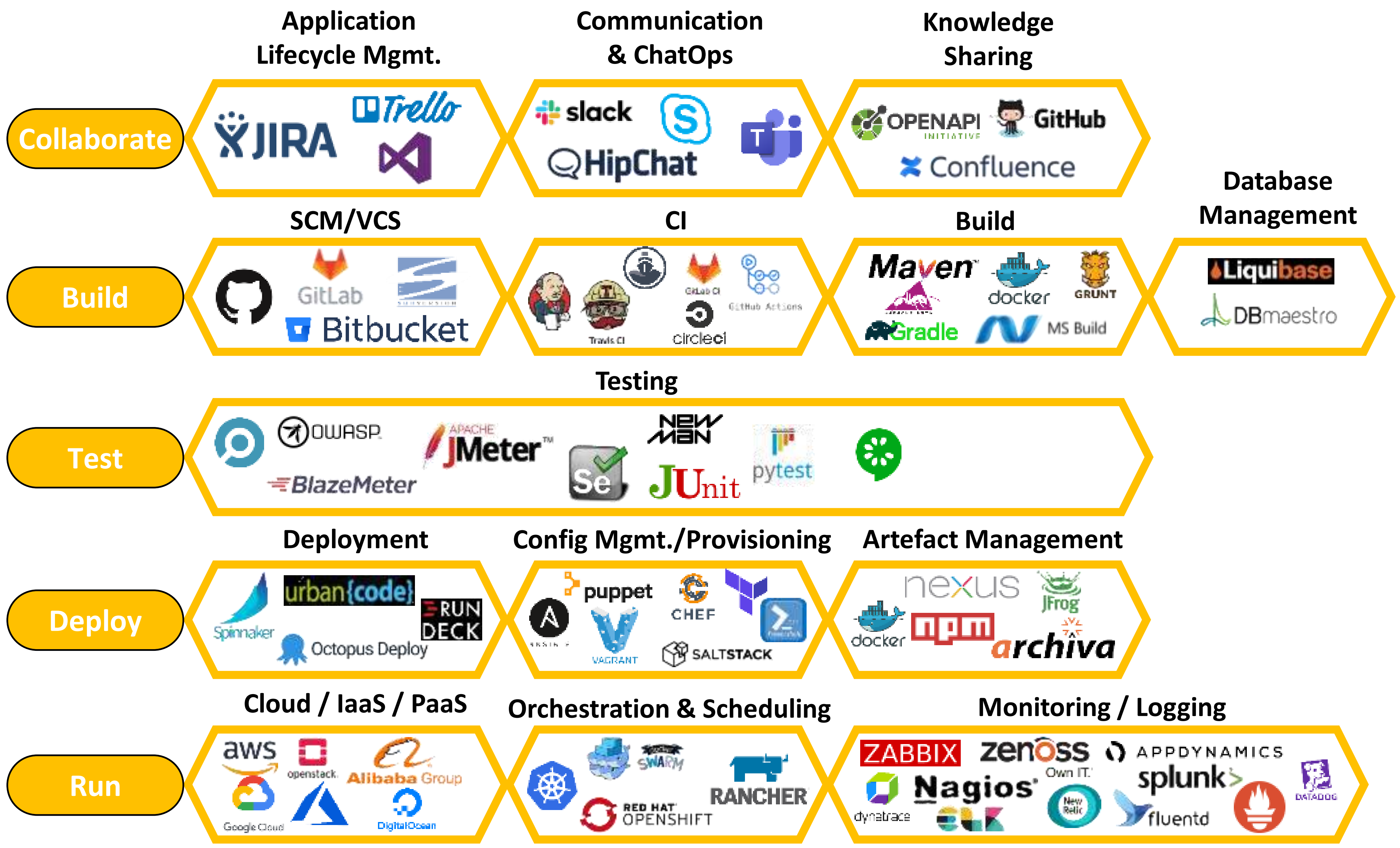
Cool tools can attract and retain

Give autonomy

Automate with existing staffs and give them a chance to learn

Take out few resources from each team, build a new virtual team for automation.

DevOps Toolsets



Introduction to Cloud Computing

What is Cloud?

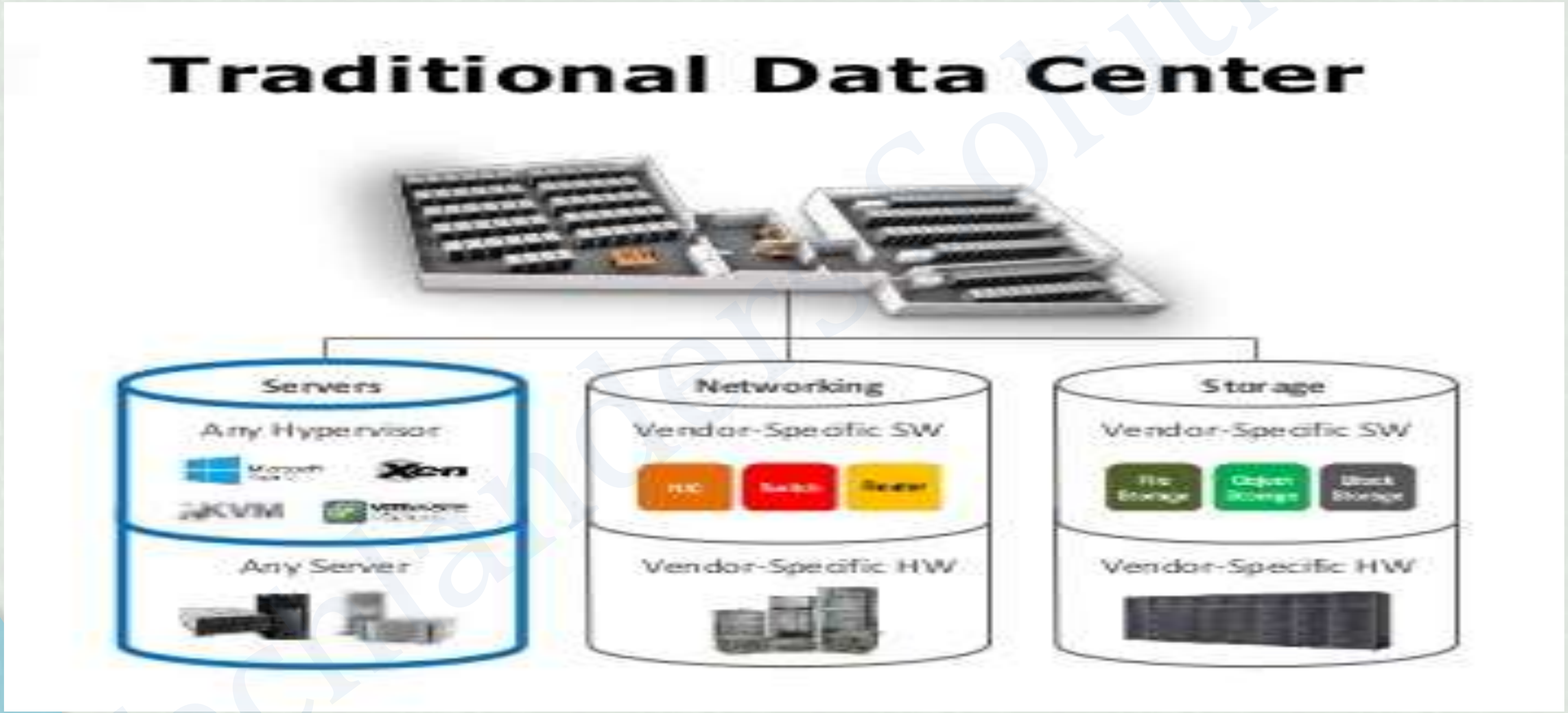
Introduction to Cloud Computing

In simple words, Cloud computing is – Placing your data on someone else's datacenter, letting them manage underline hardware Infrastructure (optionally underline Database or applications too); while having your full control on the data, and accessing that data through Internet or dedicated network.

Cloud computing is a model for enabling **universal**, **on-demand** access to a **shared pool** of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be **rapidly provisioned** and **released** with **minimal management effort** on **Pay-per-use basis**.

Free available cloud Examples:	Gmail, IRCTC, WhatsApp/Facebook
Paid available cloud Examples:	AWS, Azure(Microsoft), Oracle Cloud

Traditional DataCenters



Traditional DataCenters

- Main issues with Traditional IT Infrastructure.

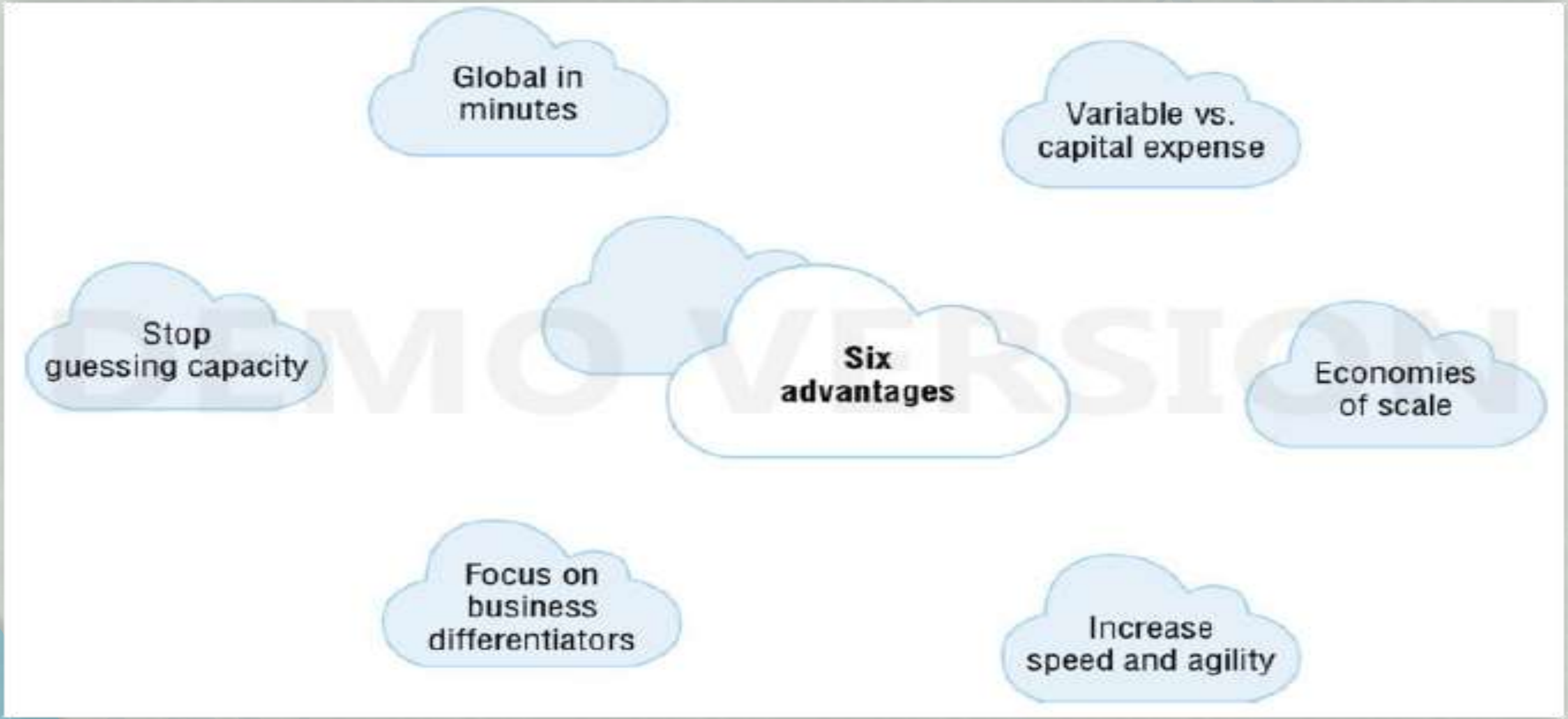
- Infrastructure is not a core business
- Hard to Scale
- Dedicated Infrastructure teams
- Dedicated Datacenters
- Dependency on vendors (servers, switches, cables etc.)
- Underutilized Resources
- High Cost
- Difficult Capacity Planning
- On-Spot demands were hard to manage
- Provisioning resources was very time consuming

Why cloud?

- To overcome all of the discussed challenges, IT infrastructure domain drifted towards Service based model which is a real “cloud computing”

- No Dedicated Datacenter
- No Different Infrastructure Teams
- Higher/Faster Scalability
- Elasticity
- Pay per use model
- Option to adopt high availability
- Better performance
- Instant provisioning
- Optimized use of resources
- On demand scaling to any extent
- No to worry about capacity planning

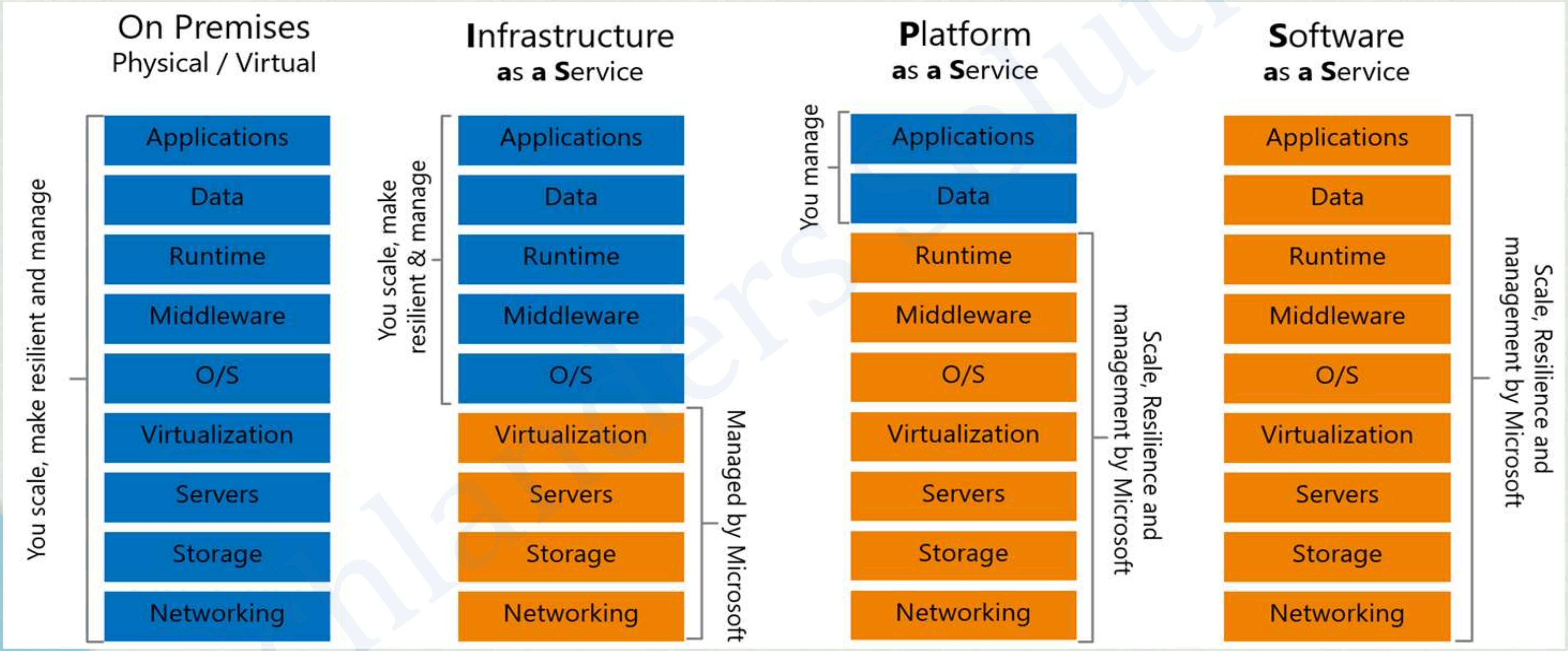
Cloud Advantages


Global in minutes
Variable vs. capital expense
Stop guessing capacity
Six advantages
Economies of scale
Focus on business differentiators
Increase speed and agility

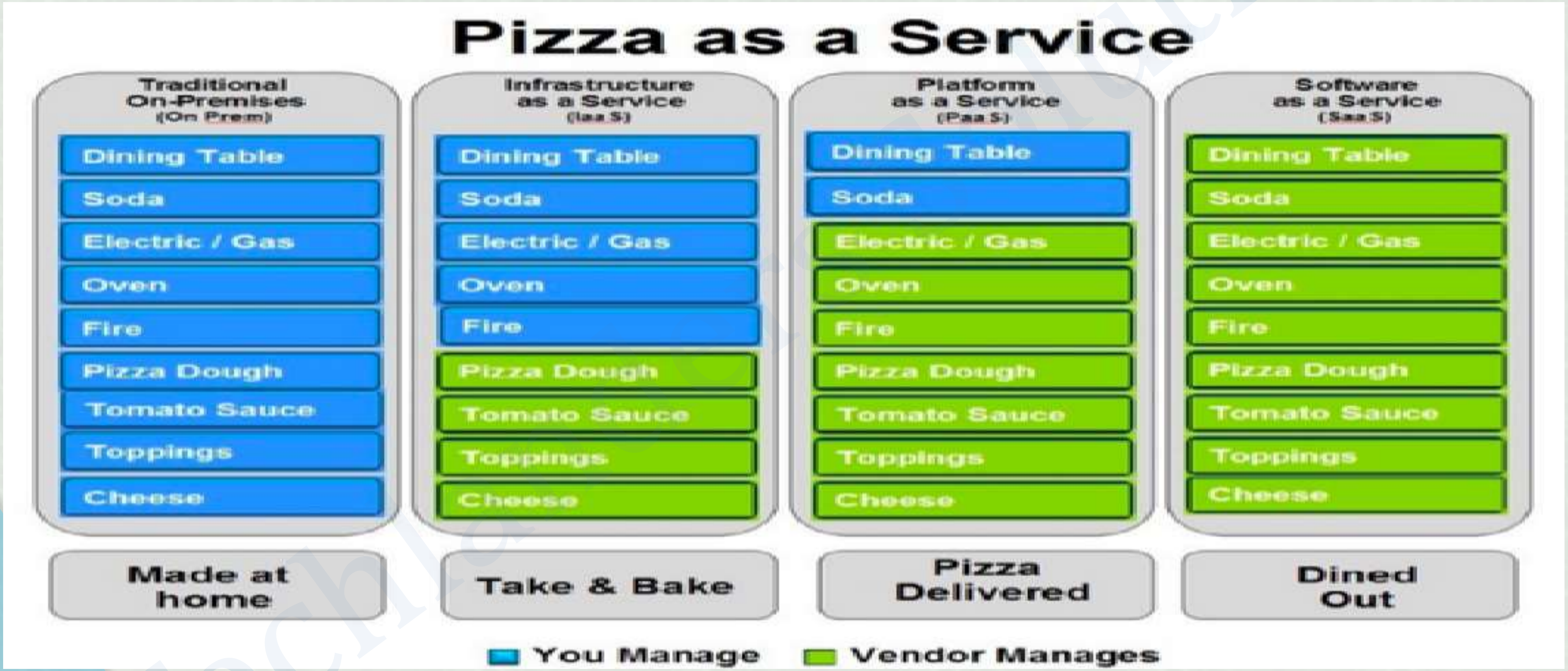
Cloud Service Models

- There are three Cloud Computing Service Models:
  - Infrastructure as a Service (IaaS)
  - Platform as a Service (PaaS)
  - Software as a Service (SaaS)

Responsibility- Who owns What?



Responsibility- Who owns What?



Pizza as a Service

Traditional On-Premises (On Prem)	Infrastructure as a Service (IaaS)	Platform as a Service (PaaS)	Software as a Service (SaaS)
🟦 Dining Table	🟦 Dining Table	🟦 Dining Table	🟩 Dining Table
🟦 Soda	🟦 Soda	🟦 Soda	🟩 Soda
🟦 Electric / Gas	🟦 Electric / Gas	🟩 Electric / Gas	🟩 Electric / Gas
🟦 Oven	🟦 Oven	🟩 Oven	🟩 Oven
🟦 Fire	🟦 Fire	🟩 Fire	🟩 Fire
🟦 Pizza Dough	🟩 Pizza Dough	🟩 Pizza Dough	🟩 Pizza Dough
🟦 Tomato Sauce	🟩 Tomato Sauce	🟩 Tomato Sauce	🟩 Tomato Sauce
🟦 Toppings	🟩 Toppings	🟩 Toppings	🟩 Toppings
🟦 Cheese	🟩 Cheese	🟩 Cheese	🟩 Cheese
Made at home	Take & Bake	Pizza Delivered	Dined Out

🟦 You Manage 🟩 Vendor Manages

Cloud Service Models - IaaS

IaaS is the most basic Cloud Service Model

It offers Underline Infrastructure for Compute, Storage and Networking

Infrastructure can be selected by customers as per their choice and Pay-per-use model.

- Examples: Bare metal servers, virtual Instances, Load balancers

IaaS - Benefits

- Drastic reduction in capital investment
- Easily Scalable
- Pay only for the used resources
- High Flexibility
- Reduced infrastructure support teams

Cloud Service Models - PaaS

Another service model, where cloud provider manages the OS & middleware part, along with IaaS

Provide capability to deploy applications on cloud infrastructure without managing underline Infra

Consumers are responsible for managing deployed applications and their environment specific configurations

- Examples: webservers and databases

PaaS - Benefits

- Includes all IaaS benefits
- No upfront licensing cost
- More reduction in Infrastructure support team
- Rapid time to market

Cloud Service Models - SaaS

SaaS deliver complete application to the consumers over the internet.

Consumers are not responsible for managing any application or underlying infrastructure.

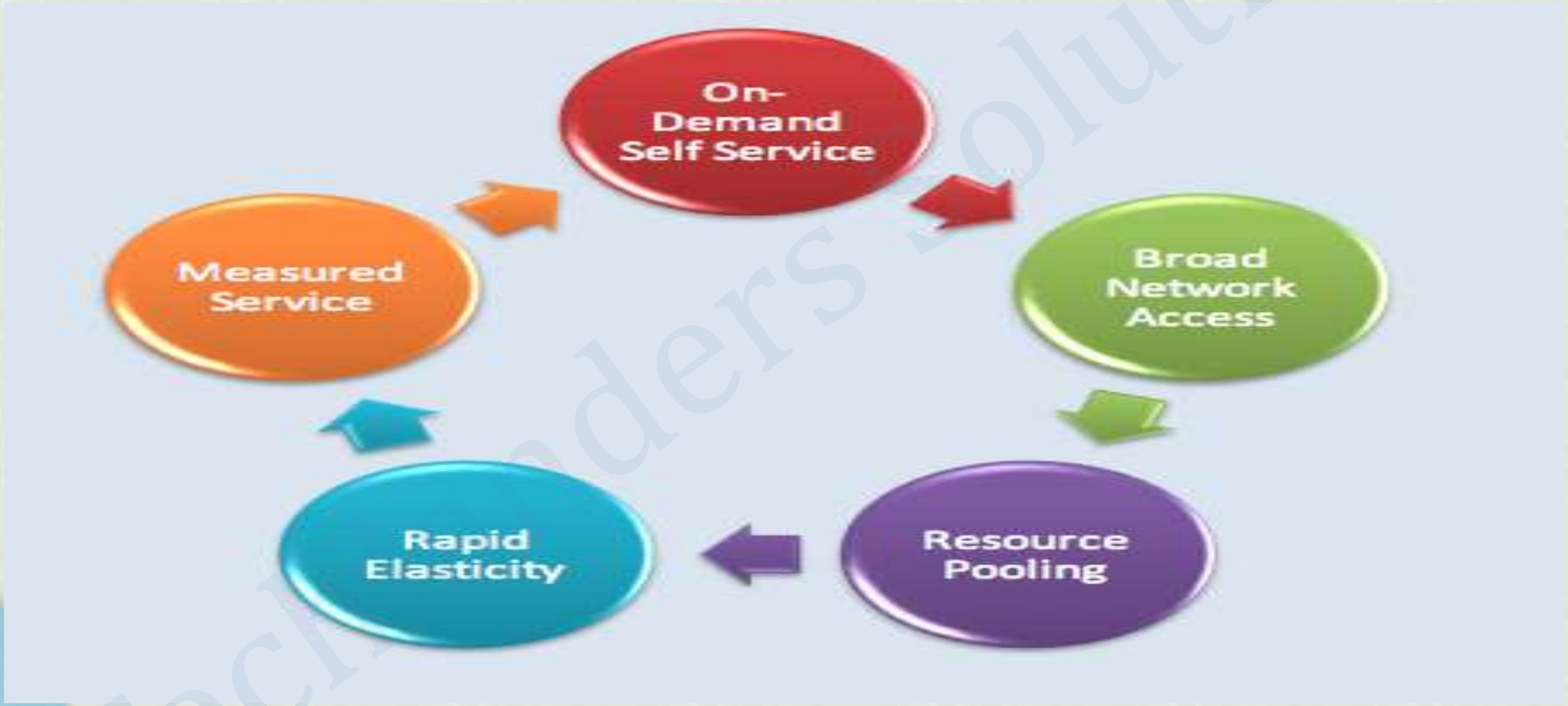
SaaS application are delivered as “one-to-many” model.

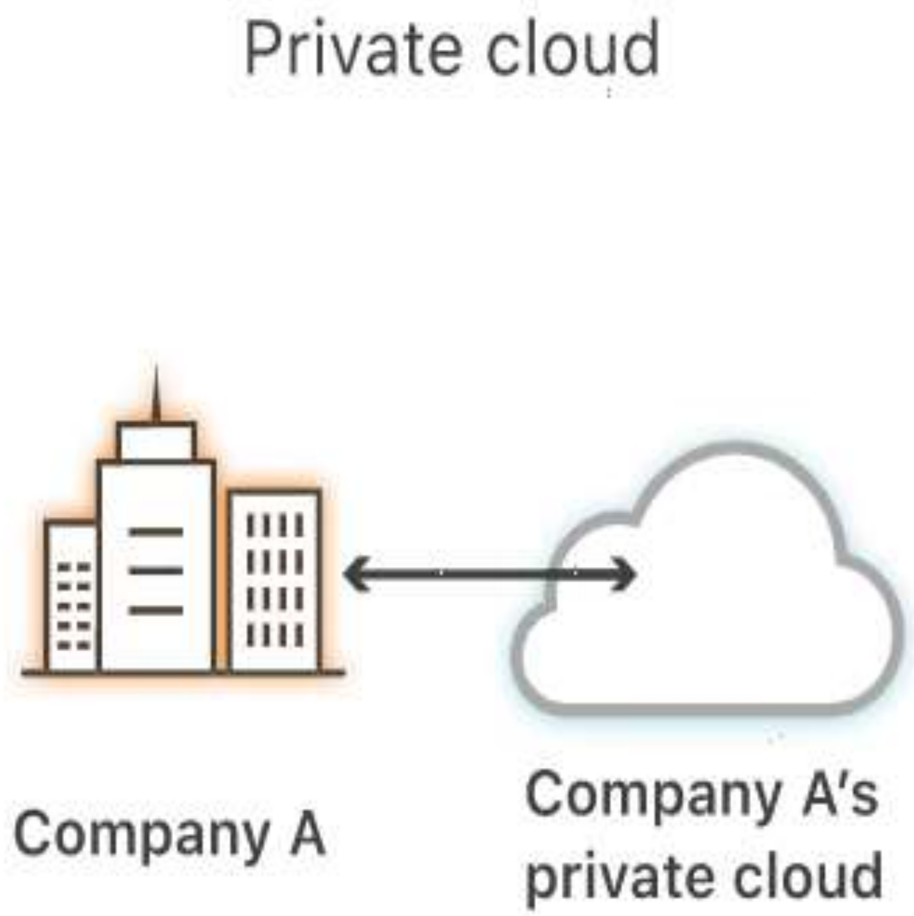
- Examples: office365, Gmail, WhatsApp, JIRA, GIT, Service Now

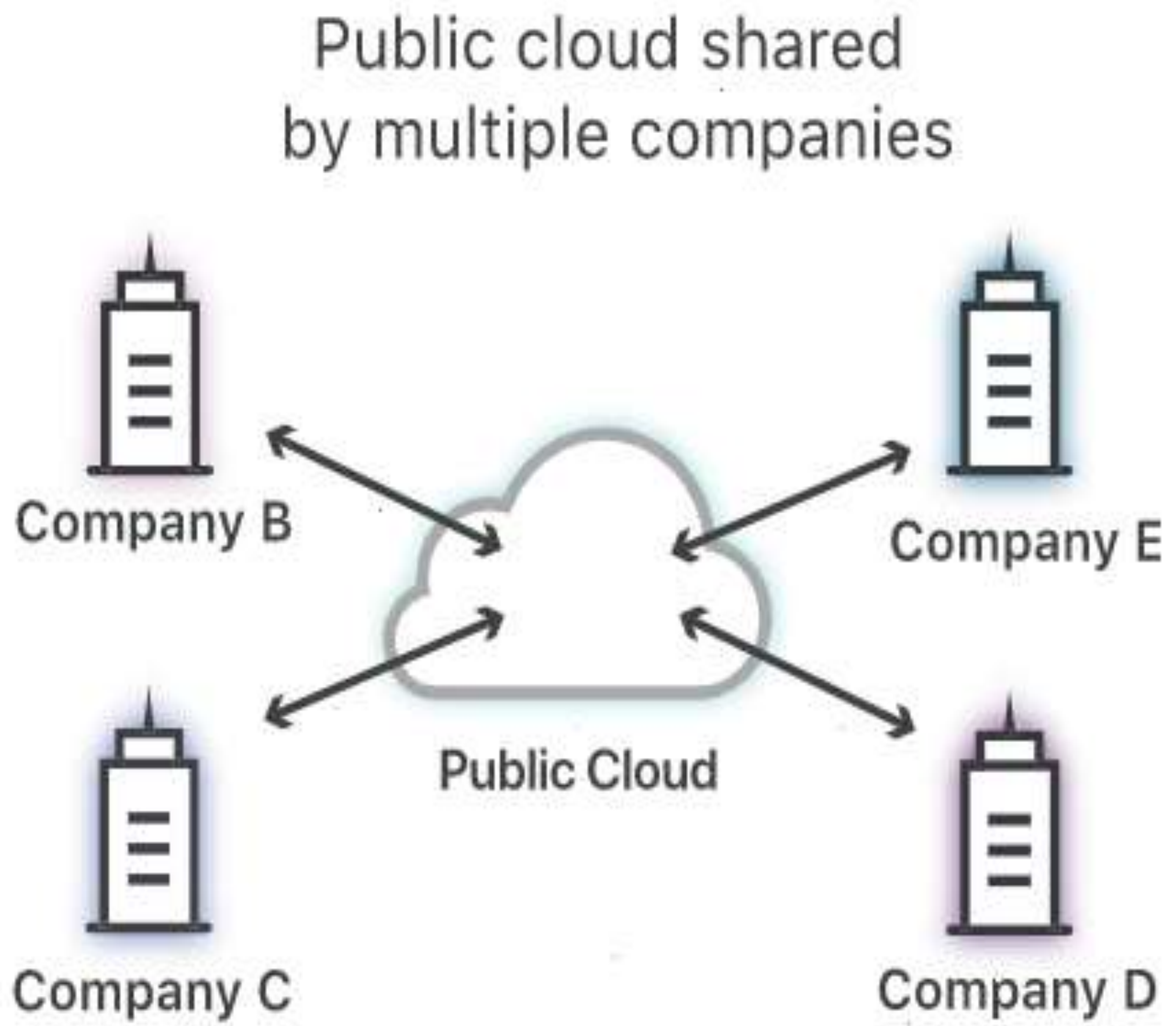
SaaS - Benefits

- Includes all discussed benefits which we get in PaaS
- Ability to access from anywhere
- Ability to access from multiple devices
- No installations and maintenance requirements
- No Application management/Licensing Required

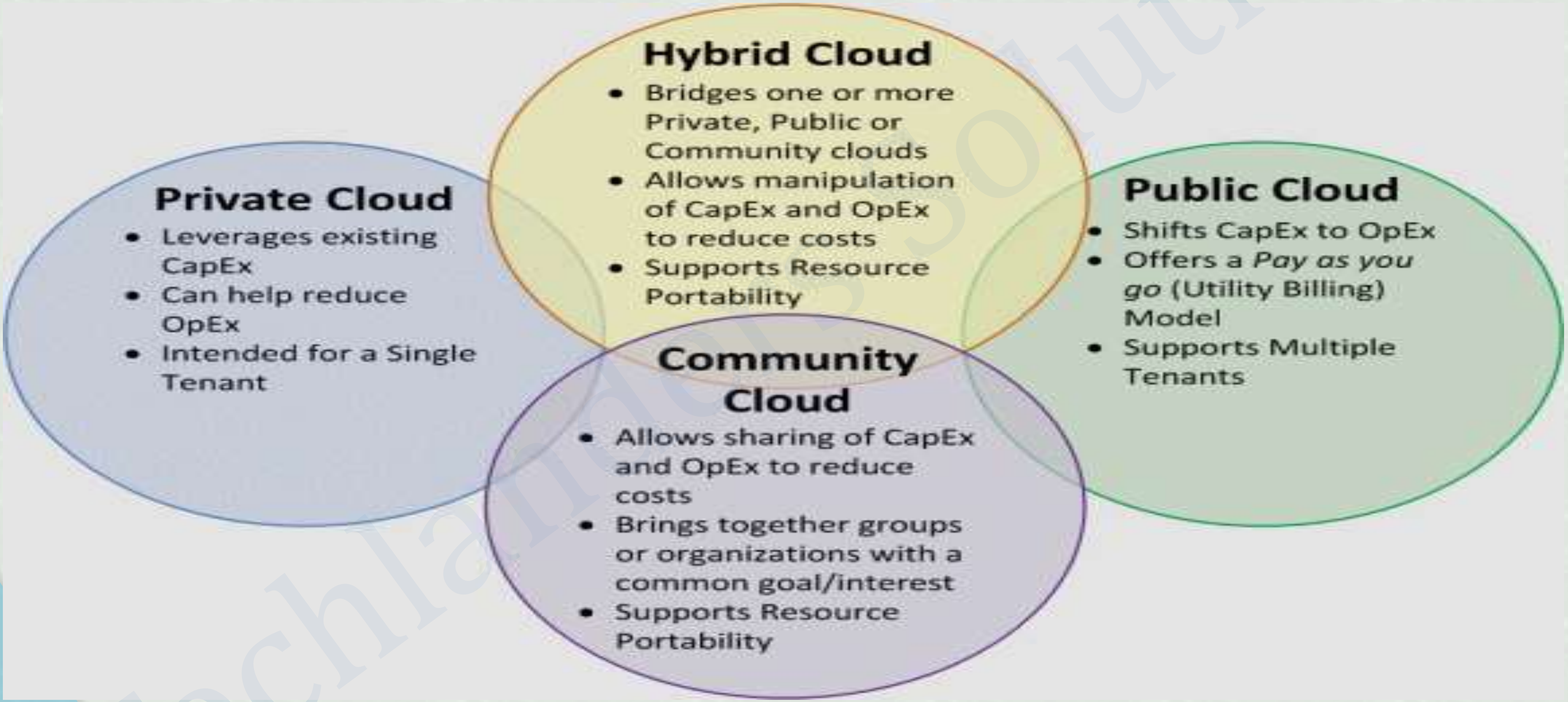
Cloud Essentials Characteristics




Private cloud
Company A
Company A's private cloud


Public cloud shared by multiple companies
Company B
Company E
Public Cloud
Company C
Company D

Cloud Deployments Types



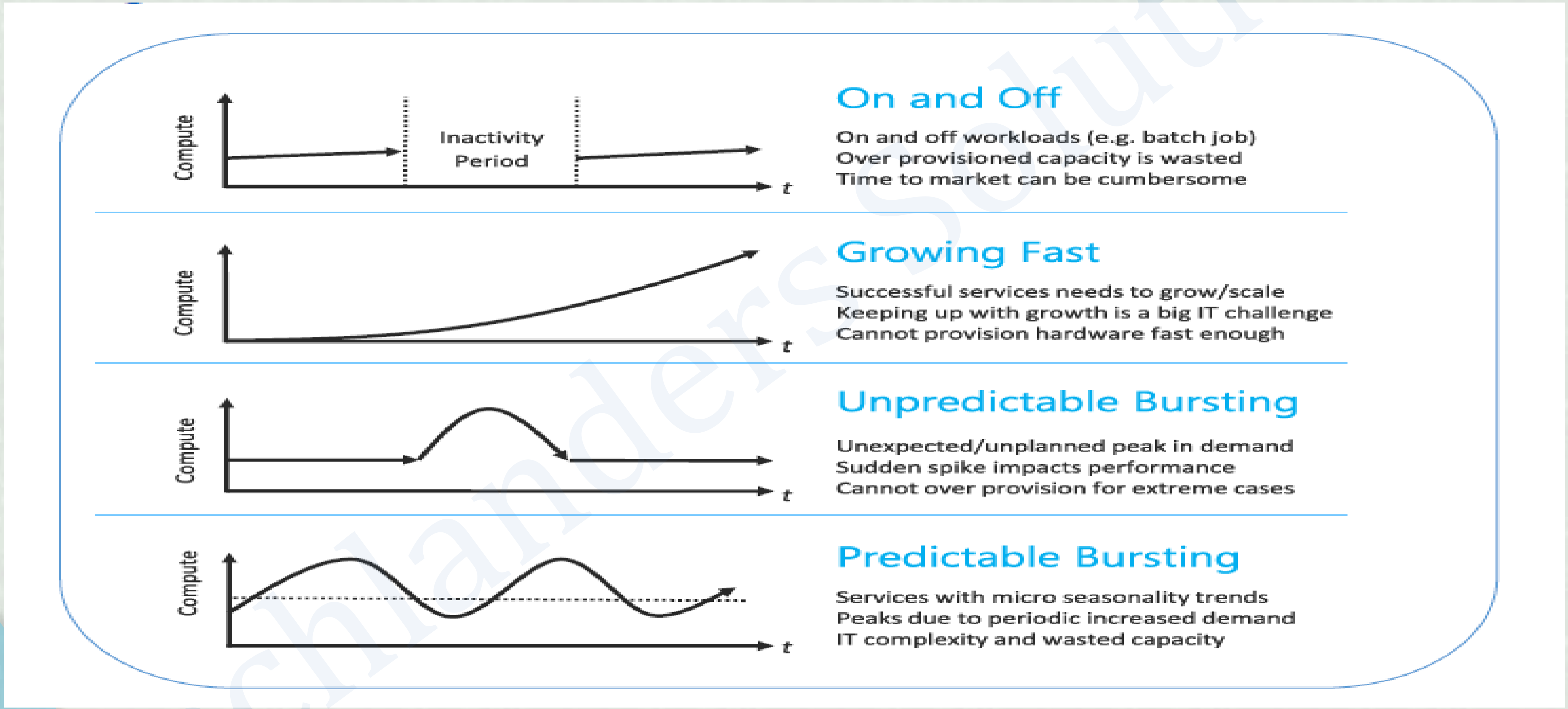
Hybrid Cloud
- Bridges one or more Private, Public or Community clouds
- Allows manipulation of CapEx and OpEx to reduce costs
- Supports Resource Portability

Private Cloud
- Leverages existing CapEx
- Can help reduce OpEx
- Intended for a Single Tenant

Public Cloud
- Shifts CapEx to OpEx
- Offers a *Pay as you go* (Utility Billing) Model
- Supports Multiple Tenants

Community Cloud
- Allows sharing of CapEx and OpEx to reduce costs
- Brings together groups or organizations with a common goal/interest
- Supports Resource Portability

Cloud's Major Use Cases



On and Off

On and off workloads (e.g. batch job)
Over provisioned capacity is wasted
Time to market can be cumbersome

Growing Fast

Successful services needs to grow/scale
Keeping up with growth is a big IT challenge
Cannot provision hardware fast enough

Unpredictable Bursting

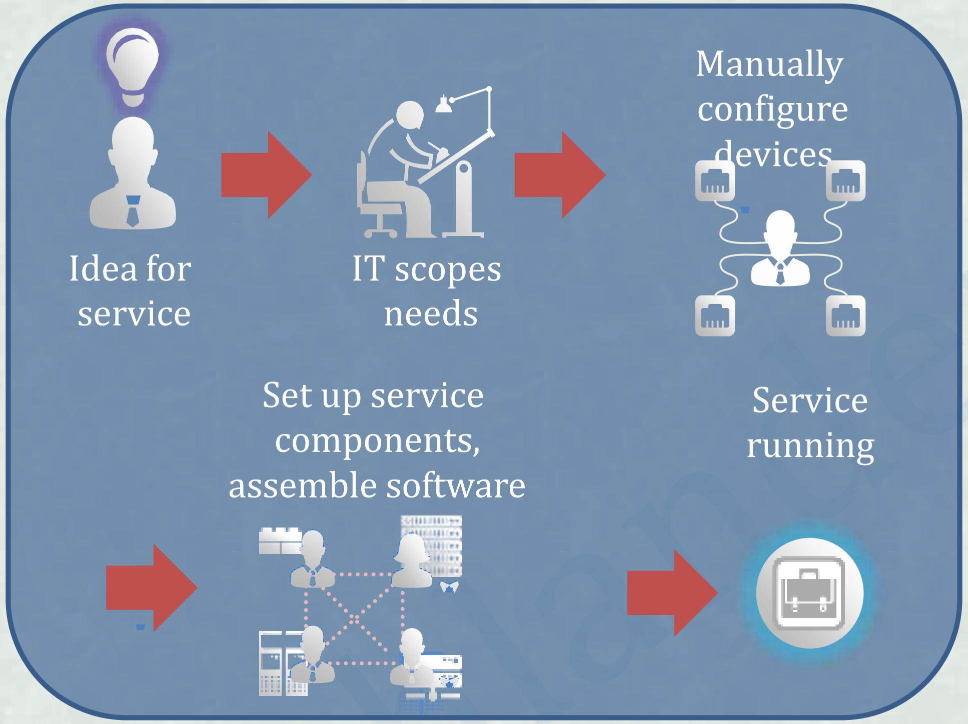
Unexpected/unplanned peak in demand
Sudden spike impacts performance
Cannot over provision for extreme cases

Predictable Bursting

Services with micro seasonality trends
Peaks due to periodic increased demand
IT complexity and wasted capacity

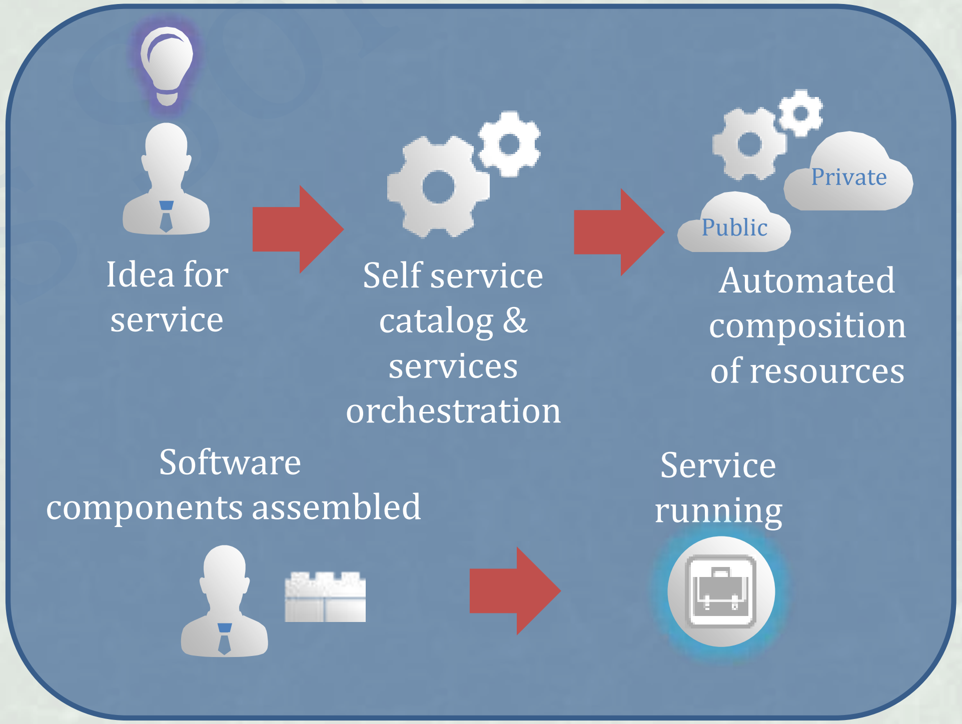
Business Impact of Cloud

Traditional Datacenter


Idea for service
IT scopes needs
Manually configure devices
Set up service components, assemble software
Service running

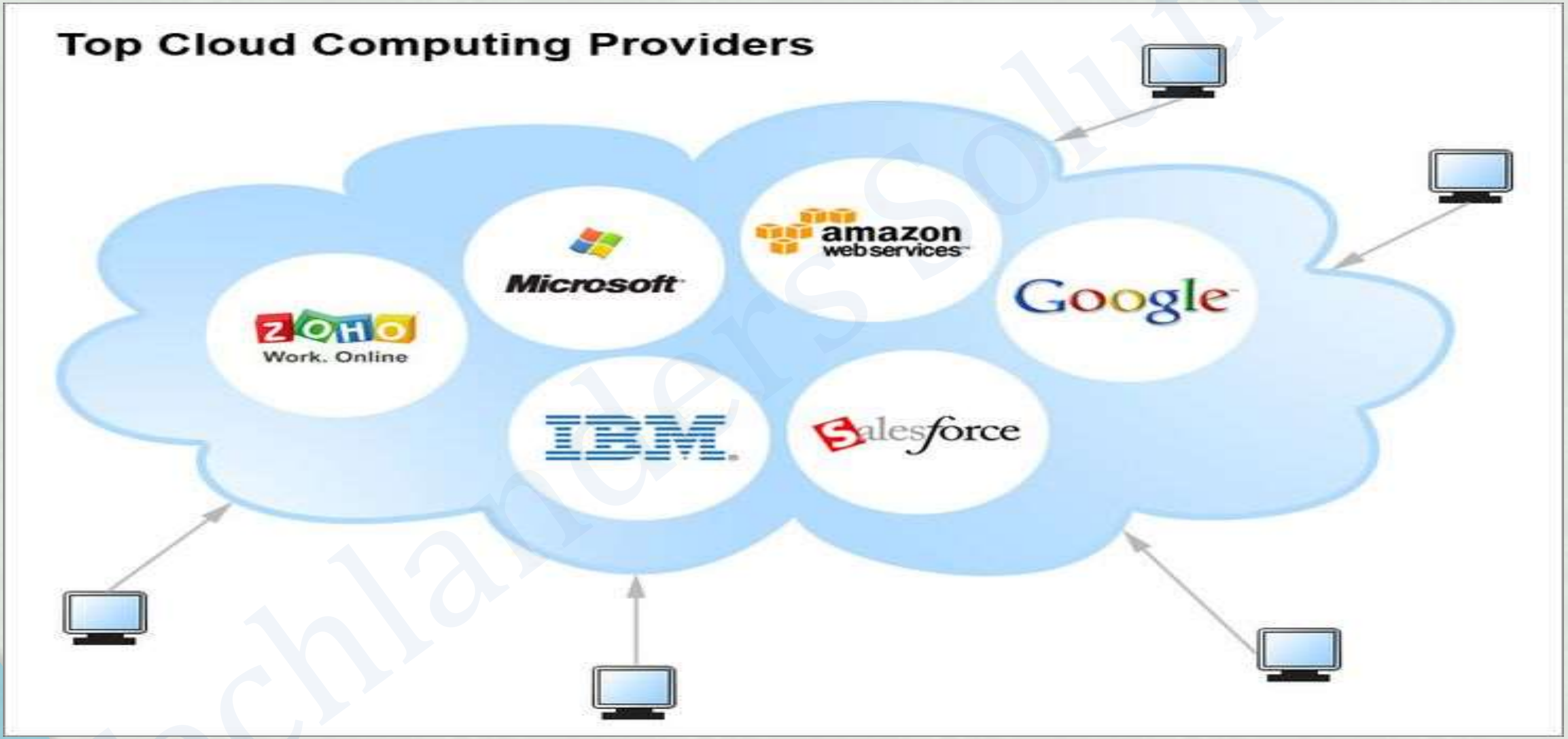
Time to Provision New Service: Months

Cloud Infrastructure


Idea for service
Self service catalog & services orchestration
Private
Public
Automated composition of resources
Software components assembled
Service running

Time to Provision New Service: Minutes

Major Cloud Vendors



Knowledge Checks

- Which Service Level (IaaS, PaaS, SaaS) provides you most control?
- What is Hybrid Cloud?
- Can two public clouds be connected?
- Connecting two public clouds, will be know as public cloud or Hybrid?
- Cloud provided Database, is a PaaS or SaaS?

AWS
(Amazon Cloud)

Amazon Web Services

- AWS (Amazon Web Services) is a group of web services (also known as cloud services) being provided by Amazon since 2006.
- AWS provides huge list of services starting from basic IT infrastructure like CPU, Storage as a service, to advance services like Database as a service, Serverless applications, IOT, Machine Learning services etc..
- Hundreds of instances can be build and use in few minutes as and when required, which saves ample amount of hardware cost for any organizations and make them efficient to focus on their core business areas.
- Currently AWS is present and providing cloud services in more than 190 countries.
- Well-known for IaaS, but now growing fast in PaaS and SaaS.

Why AWS?

- **Low Cost:** AWS offers, pay as you go pricing. AWS models are usually cheapest among other service providers in the market.
- **Instant Elasticity:** You need 1 server or 1000’s of servers, AWS has a massive infrastructure at backend to serve almost any kind of infrastructure demands, with pay for what you use policy.
- **Scalability:** Facing some resource issues, no problem within seconds you can scale up the resources and improve your application performance. This cannot be compared with traditional IT datacenters.
- **Multiple OS’s:** Choice and use any supported Operating systems.
- **Multiple Storage Options:** Choice of high I/O storage, low cost storage. All is available in AWS, use and pay what you want to use with almost any scalability.
- **Secure:** AWS is PCI DSS Level1, ISO 27001, FISMA Moderate, HIPAA, SAS 70 Type II passed. In-fact systems based on AWS are usually more secure than in-house IT infrastructure systems.

AWS Global Infrastructure

AWS Regions:

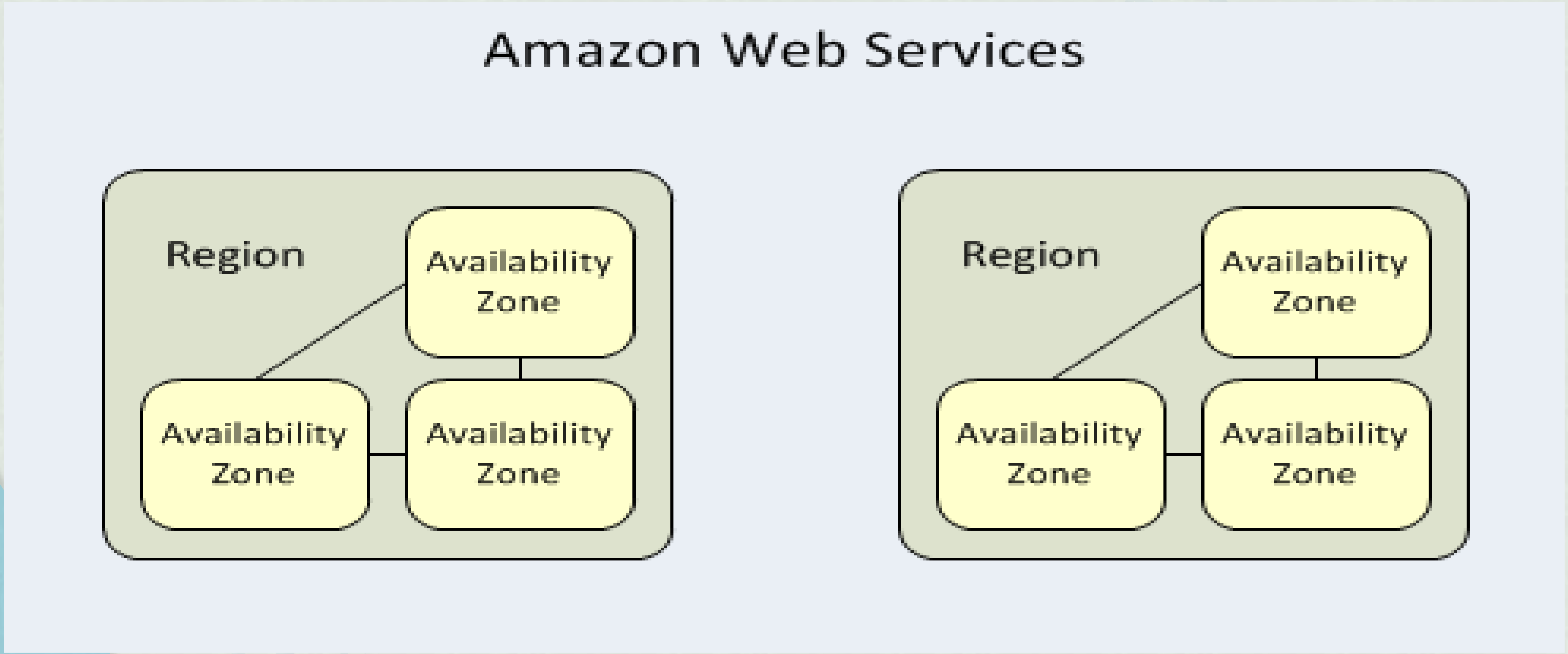
- Geographic Locations
- Consists of at least two Availability Zones(AZs)
- All of the regions are completely independent of each other with separate Power Sources, Cooling and Internet connectivity.

AWS Availability Zones

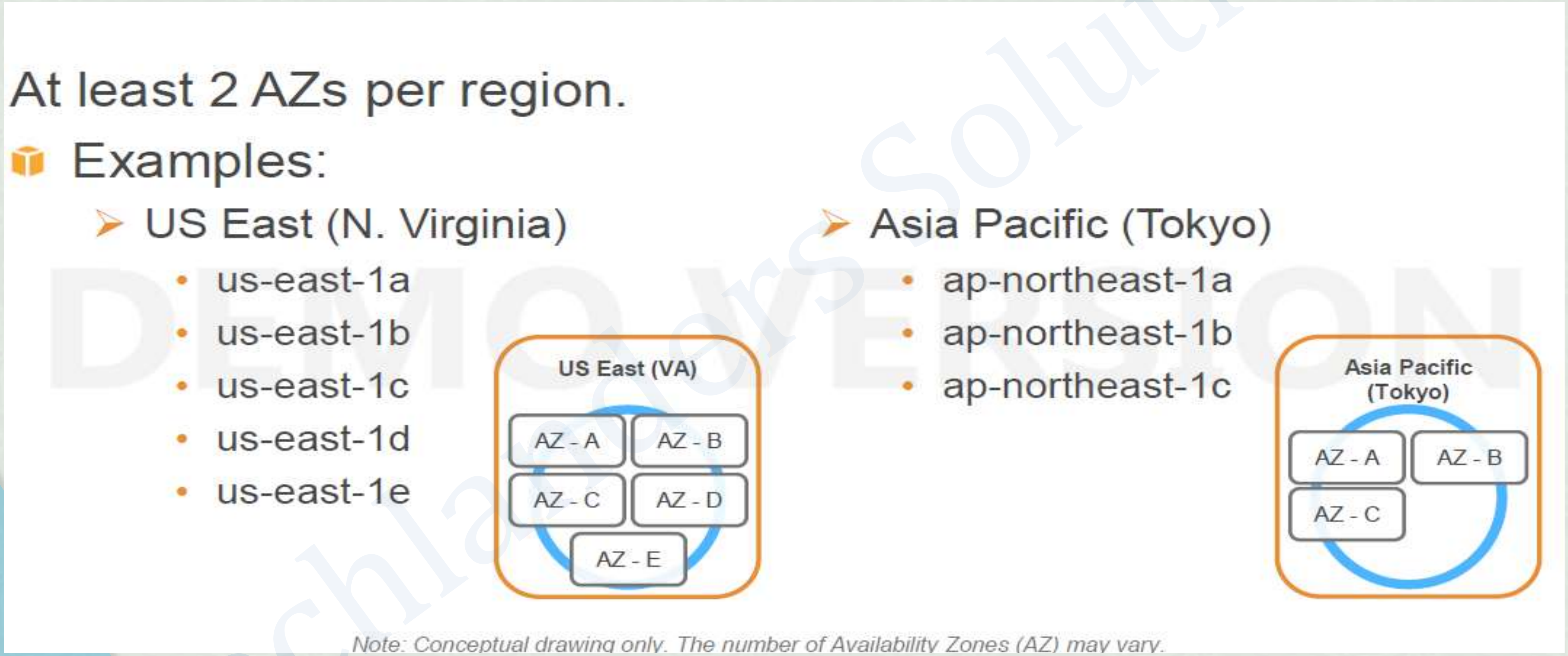
- AZ is a distinct location within a region
- Each zone is insulated (with low-latency links) from other to support single point of failures
- Each Region has minimum two AZ’s
- Most of the services/resources are replicated across AZs for HA/DR purpose.

Note: Resources aren't replicated across regions unless you do so specifically.

AWS Global Infrastructure



AWS Global Infrastructure



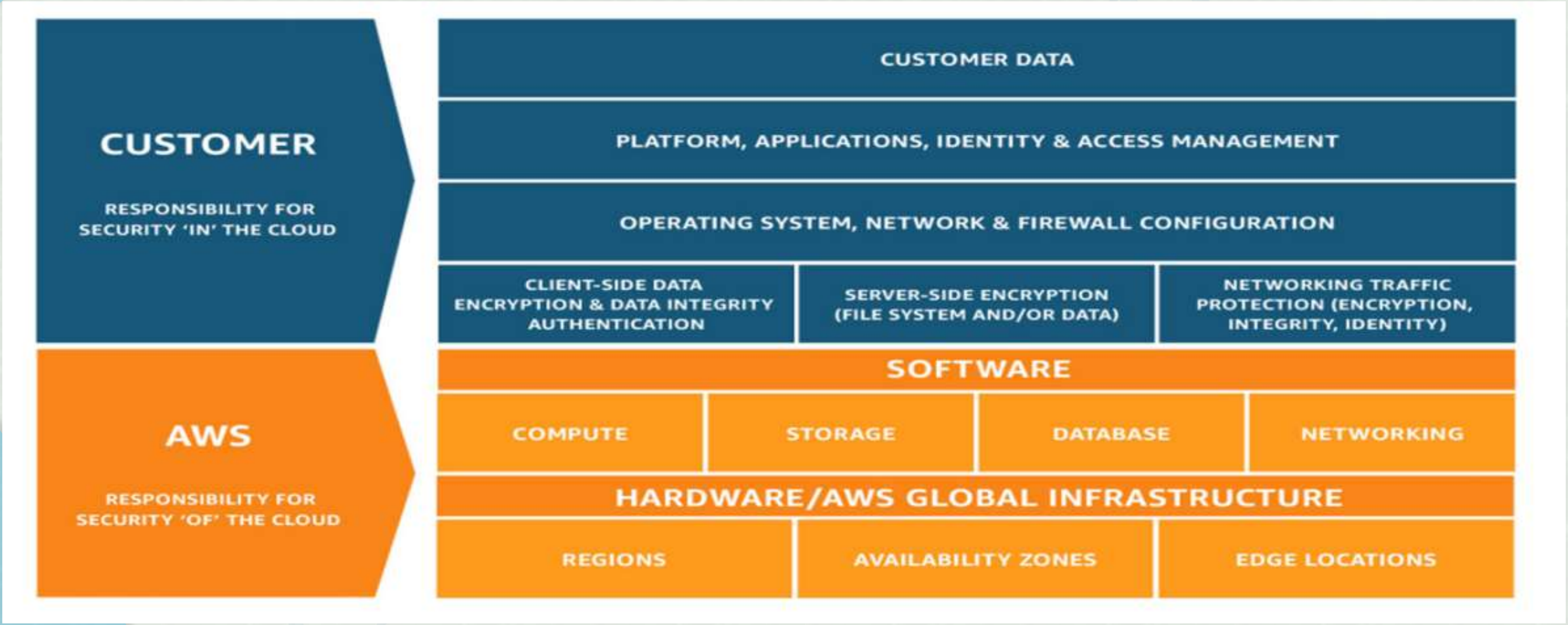
At least 2 AZs per region.

- Examples:
  - US East (N. Virginia)
    - us-east-1a
    - us-east-1b
    - us-east-1c
    - us-east-1d
    - us-east-1e
  - Asia Pacific (Tokyo)
    - ap-northeast-1a
    - ap-northeast-1b
    - ap-northeast-1c

US East (VA)
AZ - A
AZ - B
AZ - C
AZ - D
AZ - E
Asia Pacific (Tokyo)
AZ - A
AZ - B
AZ - C

Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

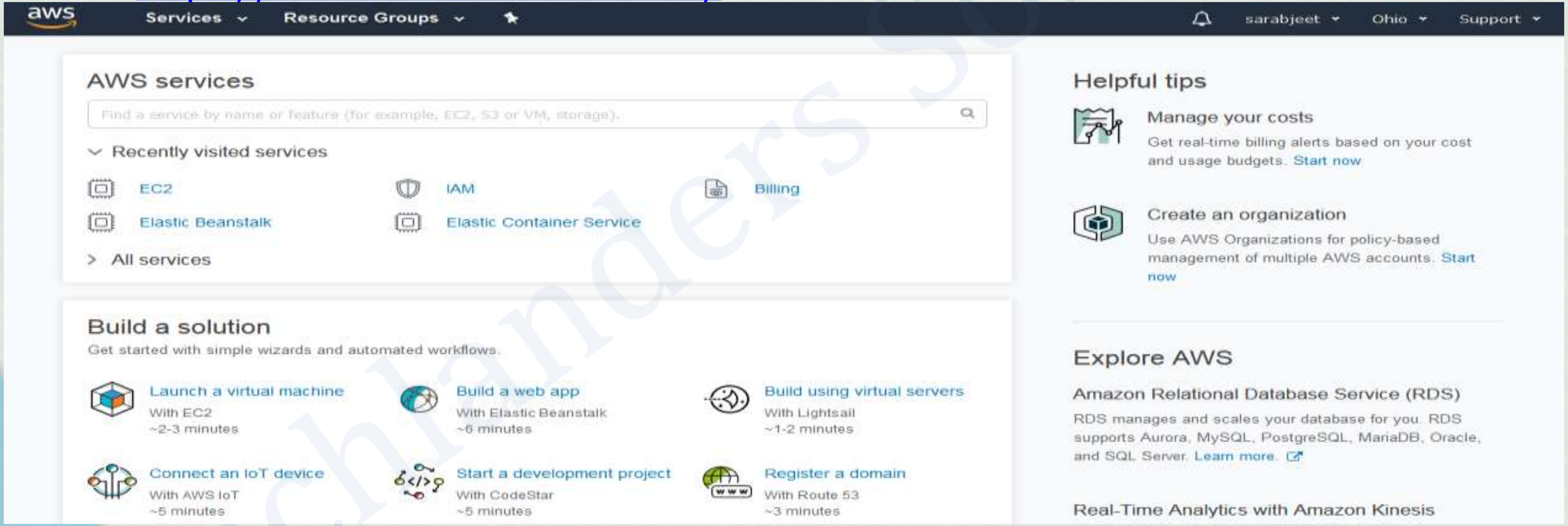
Shared Responsibility


CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

TechLanders
Delivering the future

AWS Management Console

- Simple and intuitive web-based user interface.
  - https://console.aws.amazon.com/



AWS services

Find a service by name or feature (for example, EC2, S3 or VM, storage).

Recently visited services

EC2 | IAM | Billing

Elastic Beanstalk | Elastic Container Service

All services

Build a solution

Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
~2-3 minutes

Build a web app
With Elastic Beanstalk
~6 minutes

Build using virtual servers
With Lightsail
~1-2 minutes

Connect an IoT device
With AWS IoT
~5 minutes

Start a development project
With CodeStar
~5 minutes

Register a domain
With Route 53
~3 minutes

Helpful tips

Manage your costs
Get real-time billing alerts based on your cost and usage budgets. Start now

Create an organization
Use AWS Organizations for policy-based management of multiple AWS accounts. Start now

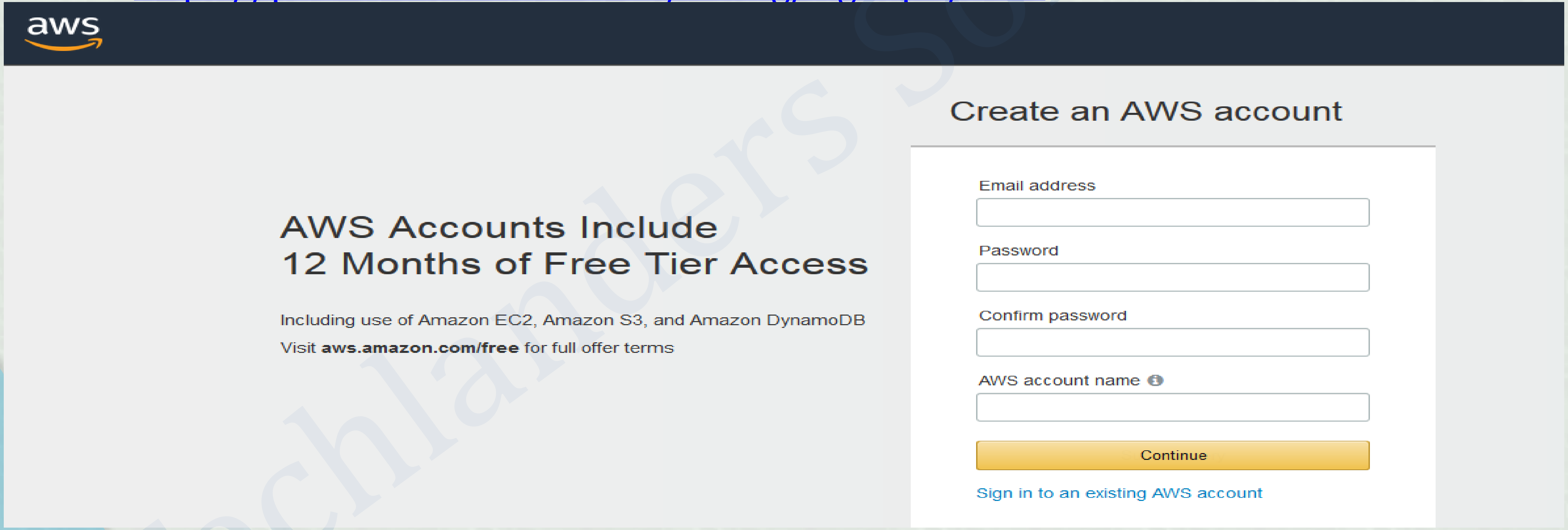
Explore AWS

Amazon Relational Database Service (RDS)
RDS manages and scales your database for you. RDS supports Aurora, MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. Learn more.

Real-Time Analytics with Amazon Kinesis

LAB 1 : AWS Signup

- Create a new account at
  - https://portal.aws.amazon.com/billing/signup#/start

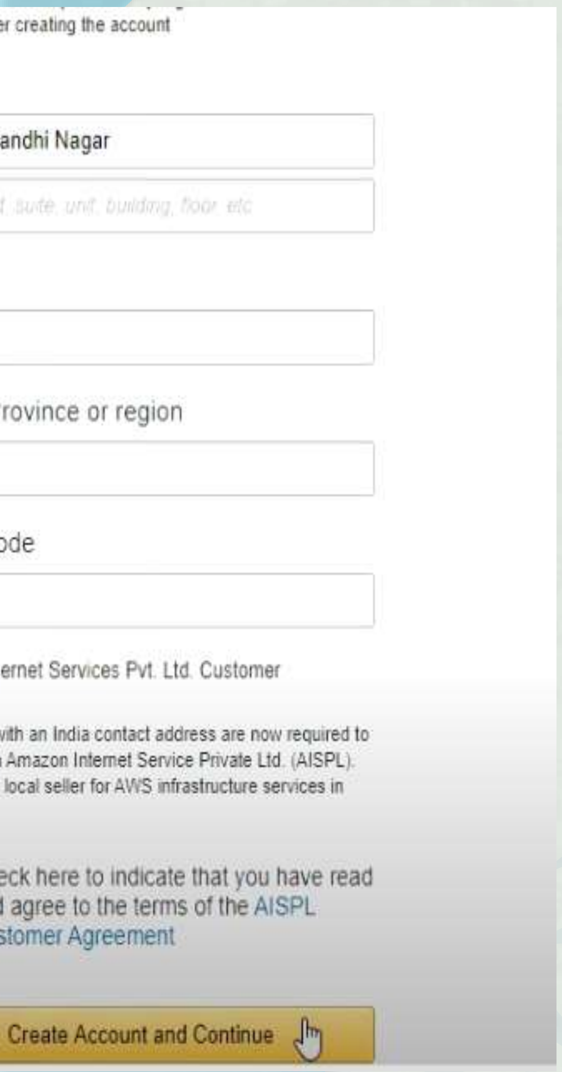


AWS Accounts Include
12 Months of Free Tier Access

Including use of Amazon EC2, Amazon S3, and Amazon DynamoDB
Visit **aws.amazon.com/free** for full offer terms

Create an AWS account

AWS SIGNUP



er creating the account

andhi Nagar

t, suite, unit, building, floor, etc

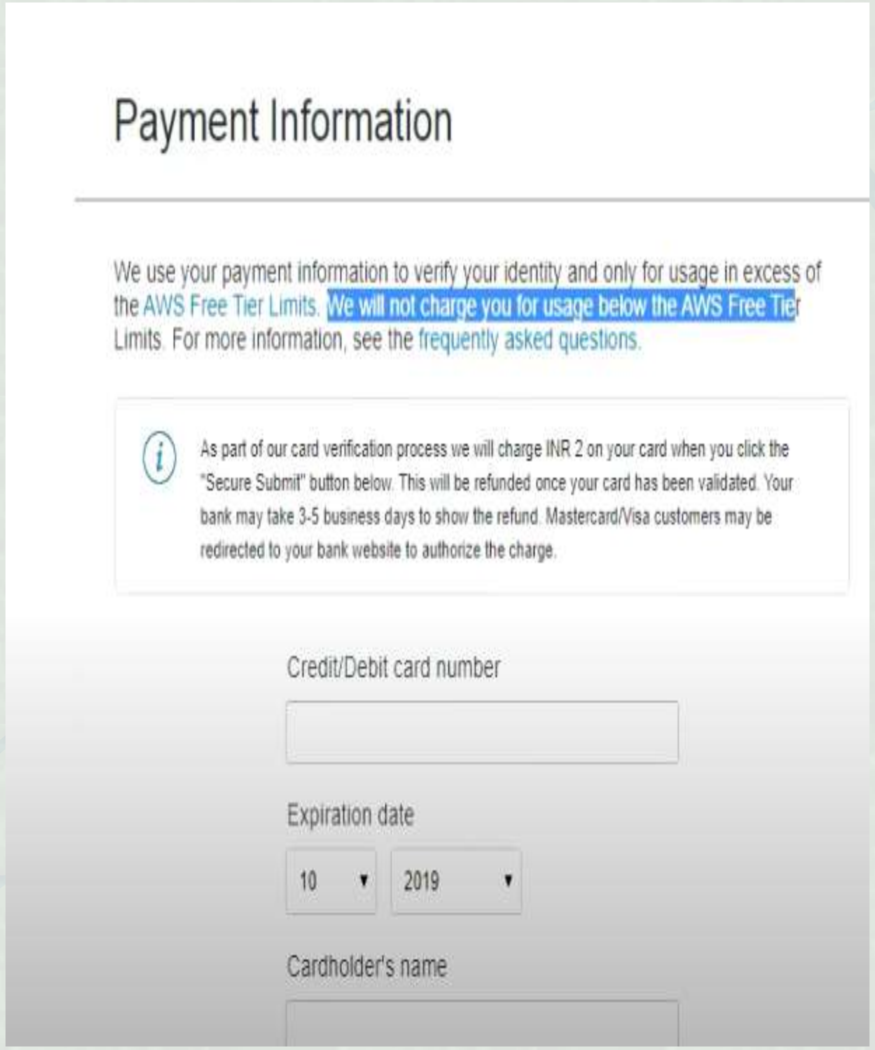
rovince or region

ode

ernet Services Pvt. Ltd. Customer

with an India contact address are now required to
Amazon Internet Service Private Ltd. (AISPL).
local seller for AWS infrastructure services in

eck here to indicate that you have read
d agree to the terms of the AISPL
stomer Agreement



Payment Information

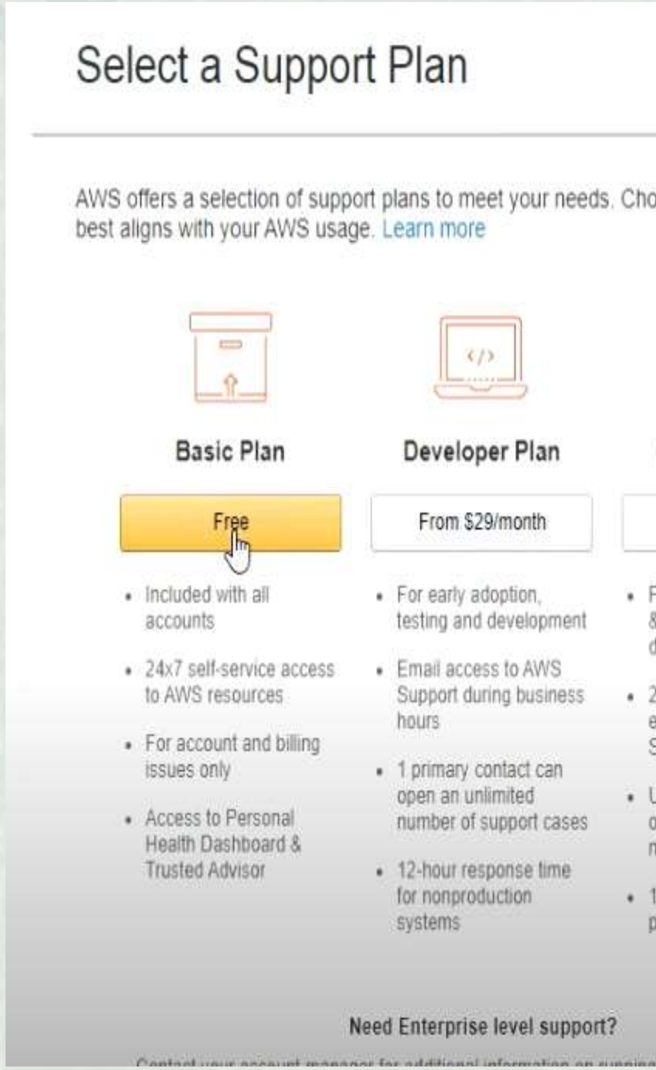
We use your payment information to verify your identity and only for usage in excess of the AWS Free Tier Limits. We will not charge you for usage below the AWS Free Tier Limits. For more information, see the frequently asked questions.

As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.

Credit/Debit card number

Expiration date: 10 / 2019

Cardholder's name



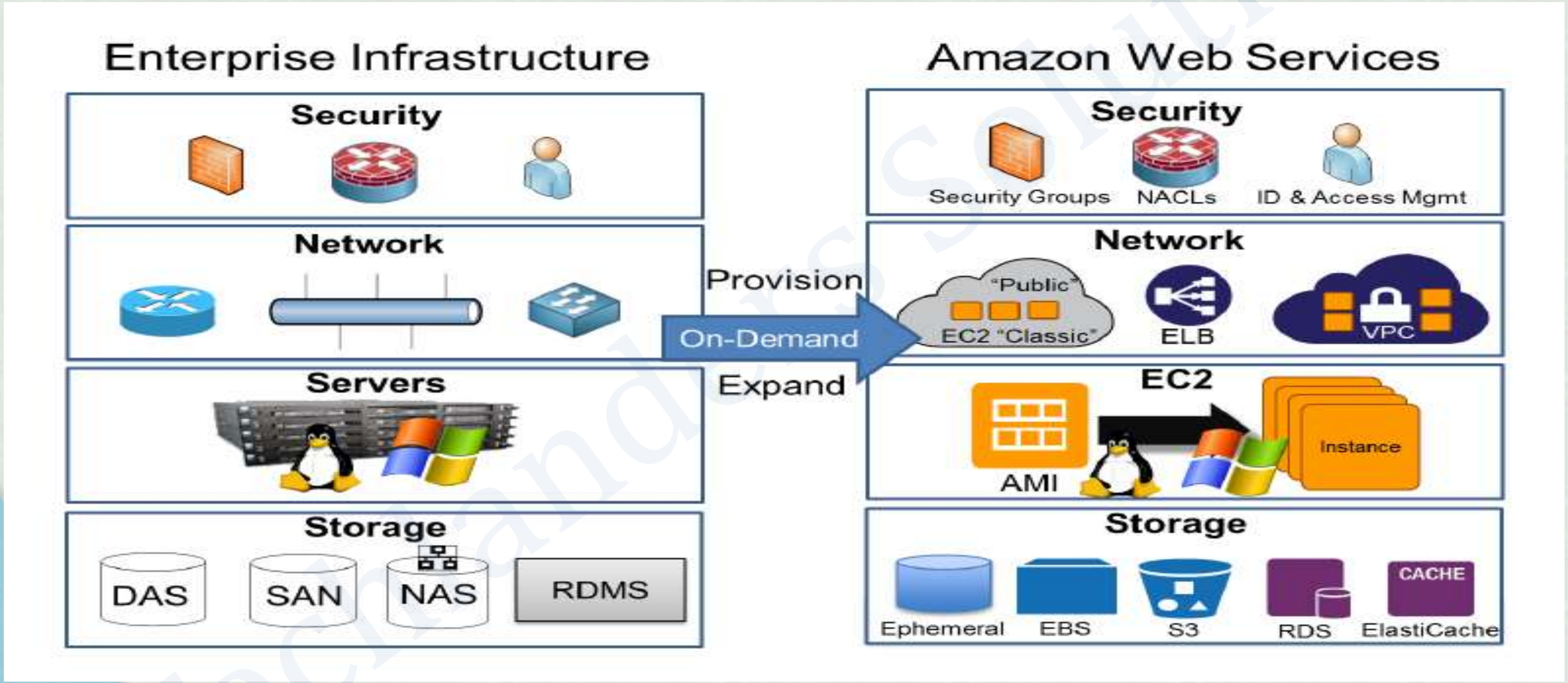
Select a Support Plan

AWS offers a selection of support plans to meet your needs. Cho
best aligns with your AWS usage. Learn more

Basic Plan	Developer Plan
Free	From $29/month
• Included with all accounts • 24x7 self-service access to AWS resources • For account and billing issues only • Access to Personal Health Dashboard & Trusted Advisor	• For early adoption, testing and development • Email access to AWS Support during business hours • 1 primary contact can open an unlimited number of support cases • 12-hour response time for nonproduction systems

Need Enterprise level support?

AWS Core Infrastructure Services



TechLanders
Delivering the future

AWS Security

- **Physical Security:**
- 24/7 trained security staff
- AWS data centers in nondescript and undisclosed facilities
- Two-factor authentication for authorized staff
- Authorization for data center access
- Multiple approval based change process

Amazon Resource Names (ARNs)

Amazon Resource Names (ARNs) uniquely identify AWS resources.

We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, API calls etc.

ARN have a specific format:

arn:partition:service:region:account-id:resourcetype/resource

- IAM user name
  arn:aws:iam::123456789012:user/David
- IAM instance id:
  arn:aws:ec2:*region*:*account-id*:dedicated-host/*host_id*
  Eg. arn:aws:ec2:us-east-1:123456789012:dedicated-host/h-12345678

AWS Access Credentials

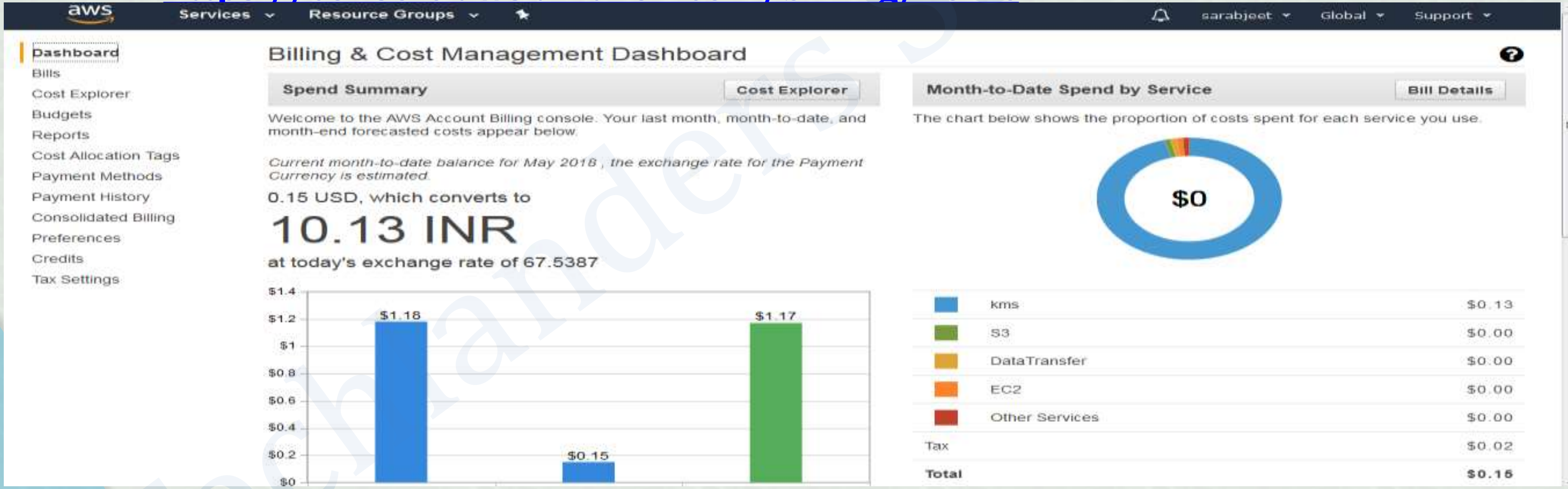
AWS resources can be access using several authentication methods:

- IAM User-id / Password
- Account ID/ AK (Access Key)/ SK (Security Key)
- Certificates
- Key pairs

AWS Billing

AWS Billings history, Previous payments, Current month cost, Budget fixing, Setting usage alarms etc, can be managed from AWS billing page :

https://console.aws.amazon.com/billing/home



Billing & Cost Management Dashboard

Spend Summary

Welcome to the AWS Account Billing console. Your last month, month-to-date, and month-end forecasted costs appear below.

Current month-to-date balance for May 2018 , the exchange rate for the Payment Currency is estimated.

0.15 USD, which converts to

10.13 INR

at today's exchange rate of 67.5387

$1.18	
$0.15	
$1.17	

Month-to-Date Spend by Service

The chart below shows the proportion of costs spent for each service you use.

🟦	kms	$0.13
🟩	S3	$0.00
🟨	DataTransfer	$0.00
🟧	EC2	$0.00
🟥	Other Services	$0.00
	Tax	$0.02
	Total	**$0.15**

AWS Pricing calculators

Simple Monthly Calculator:
You can Estimate your expected monthly bill using Simple Monthly Calculator.
http://calculator.s3.amazonaws.com/index.html

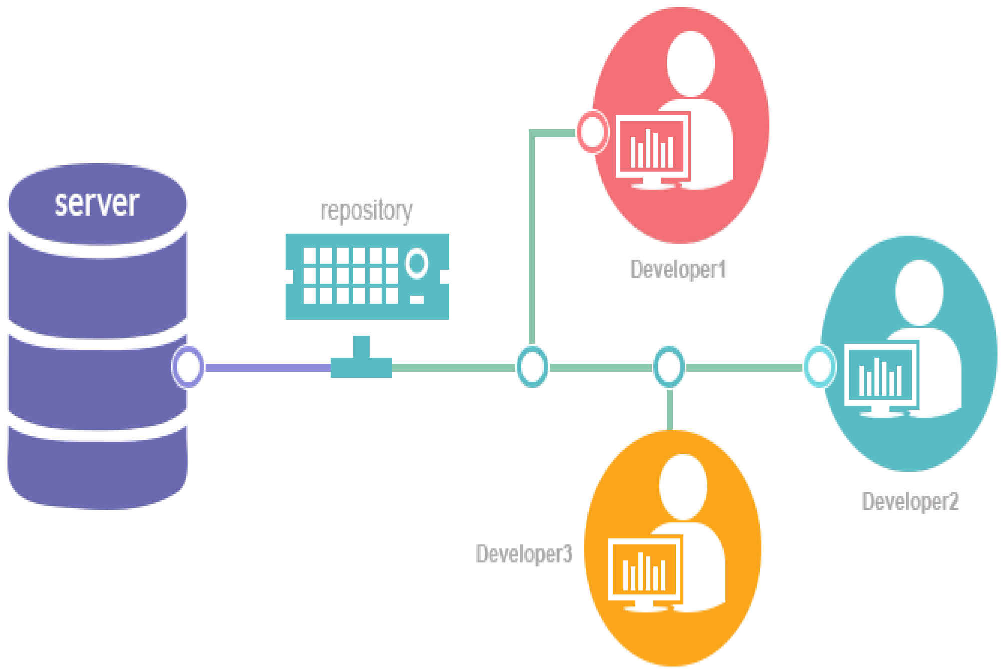
TCO Calculator:
You can Quickly compare the total cost of ownership (TCO) of your **on-premises infrastructure with a comparable AWS deployment** using TCO Calculator and estimate savings you can realize by moving to AWS. https://awstcocalculator.com/#

Cost Explorer:
With Cost Explorer, you can track your actual account usage and bill, at any time using the billing portal. You can view data for up to the last 13 months, forecast how much you are likely to spend for the next three months, and get recommendations for what Reserved Instances to purchase.
https://console.aws.amazon.com/billing/home#/costexplorer

Version Control Systems with GIT

What is The Need of Version Control System

How Version Control System Will Work ?



Basic Terminology of Version Control System

- Working Directory
- Repository
- Commit
- Checkout

Advantage of Version Control System

- With a distributed system, you can work on your copy of the code without having to worry about ongoing work on the same code by others.
- Identify the ownership of changes
- you can **sync your repositories among yourselves**, bypassing the central location.
- **Managing access is easier** in distributed systems.

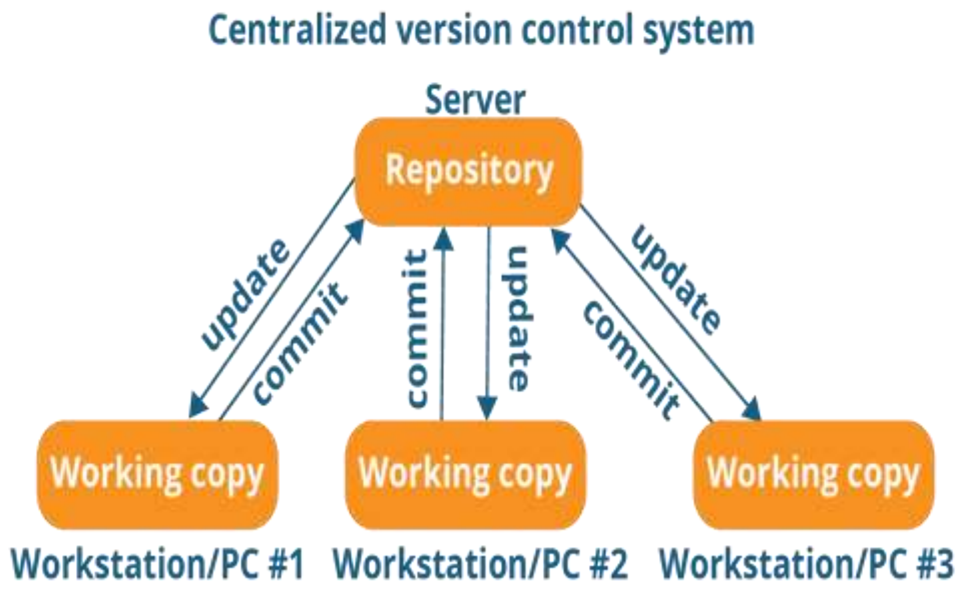
Types of Version Control System

There are two types of version control systems (VCS).

- Centralized version control systems (CVCS)
- Distributed version control systems (DVCS).

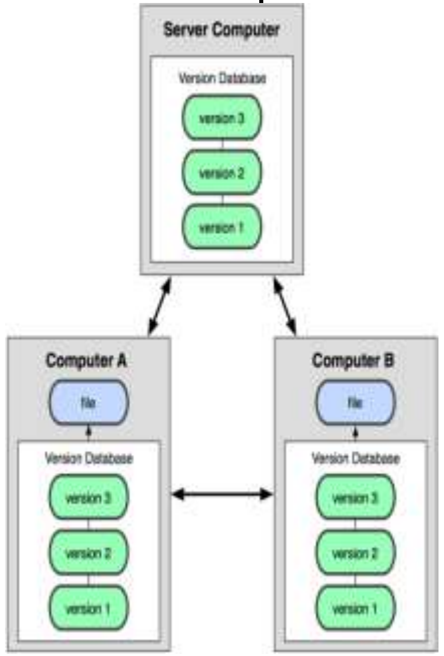
Centralized version control systems (CVCS)

Centralized systems have a copy of the project hosted on a centralized server, to which everyone connects to in order to make changes. Here, the “first come, first served” principle is adopted: if you’re the first to submit a change to a file, your code will be accepted.



Distributed version control systems (DVCS)

In a distributed system, every developer has a copy of the entire project. Developers can make changes to their copy of the project without connecting to any centralized server, and without affecting the copies of other developers. Later, the changes can be synchronized between the various copies.



Introduction of Git

Git History

Linus uses BitKeeper to manage Linux code

Ran into BitKeeper licensing issue

Liked functionality

Looked at CVS as how not to do things

- April 5, 2005 - Linus sends out email showing first version
- June 15, 2005 - Git used for Linux version control

Why Git?

- **Branching:** gives developers a great flexibility to work on a replica of master branch.
- **Distributed Architecture:** The main advantage of DVCS is **"no requirement of network connections to central repository"** while development of a product.
- **Open-Source:** Free to use.
- **Integration with CI:** Gives faster product life cycle with even faster minor changes.

What is Git

- Git is a distributed version control system that is used by developers to manage changes to a codebase.
- Git uses a branching model that allows developers to work on different features and changes to the codebase without affecting the main branch.
- Git is fast, scalable, and efficient, making it an essential tool for software development.
- Git is used by millions of developers and companies worldwide, and it can be used with a wide range of programming languages and operating systems.
- Git provides a range of tools and commands for managing changes to the codebase, including committing changes, branching, merging, and resolving conflicts.

Git Installation

Open a terminal: You can use the shortcut Ctrl + Alt + T on most distributions.

Update package lists: Run the following command to update the package lists and ensure you are installing the latest version of Git:

- Update The system

```
sudo apt update –y  / sudo yum update -y
```

- Install Git: Run the following command to install Git on your system:

```
sudo apt install git / sudo yum install git -y
```

- Verify the installation: After the installation is complete, you can verify it by checking the Git version:

```
git --version
```

Git Commands

- git –version   // to check the version

// set the global user name and email

- [root@techlanders ~]# git config --global user.email “sandeep@techlanders.com”
- [root@techlanders ~]# git config --global user.name “sandeep”
- [root@techlanders ~]# git config --global -l
- user.name=sandeep
- user.email= sandeep@techlanders.com

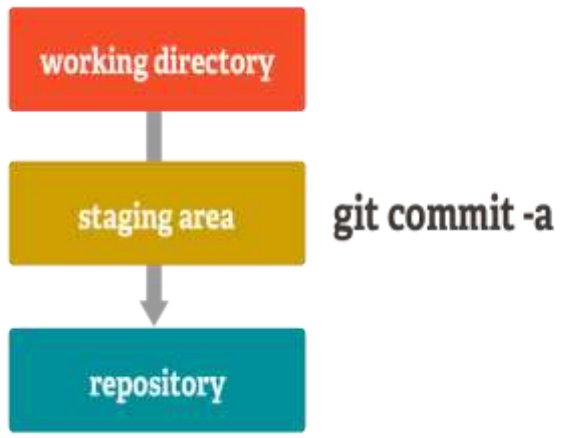
****

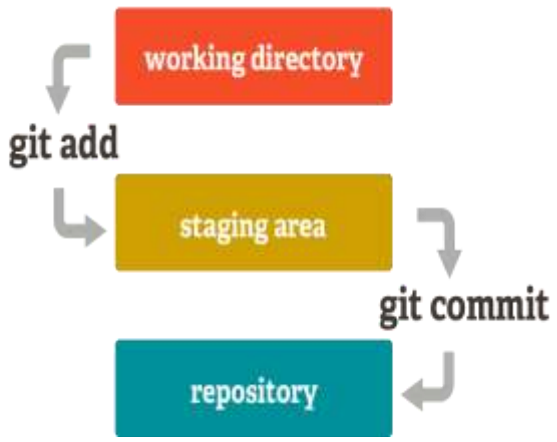
//Initializing a repo

- [root@master git]# mkdir /Repo1
- [root@master git]# cd /Repo1/
- [root@master Repo1]# git init
- Initialized empty Git repository in /Repo1/.git/
- [root@master Repo1]#

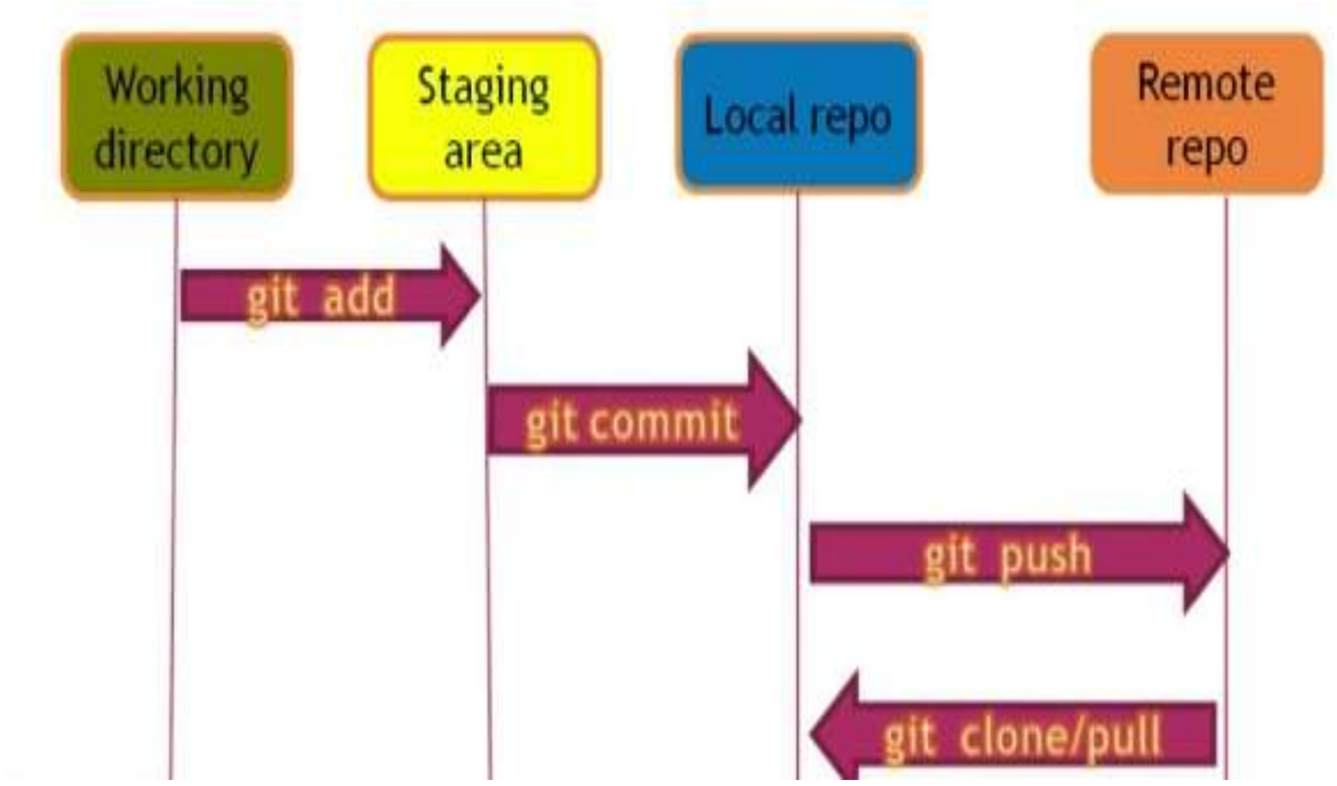
Staging Area

- Unlike the other systems, Git has something called the "staging area" or "index". This is an intermediate area where commits can be formatted and reviewed before completing the commit.

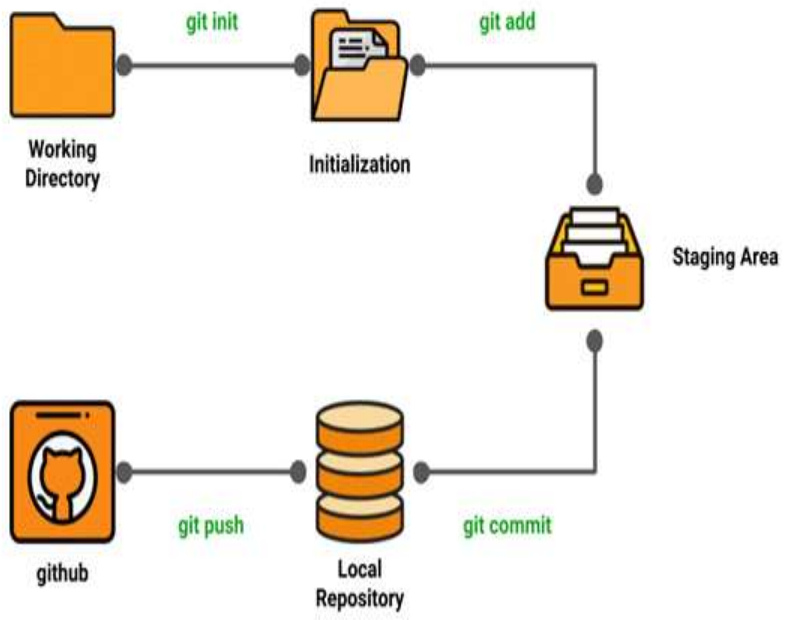




GIT Architecture



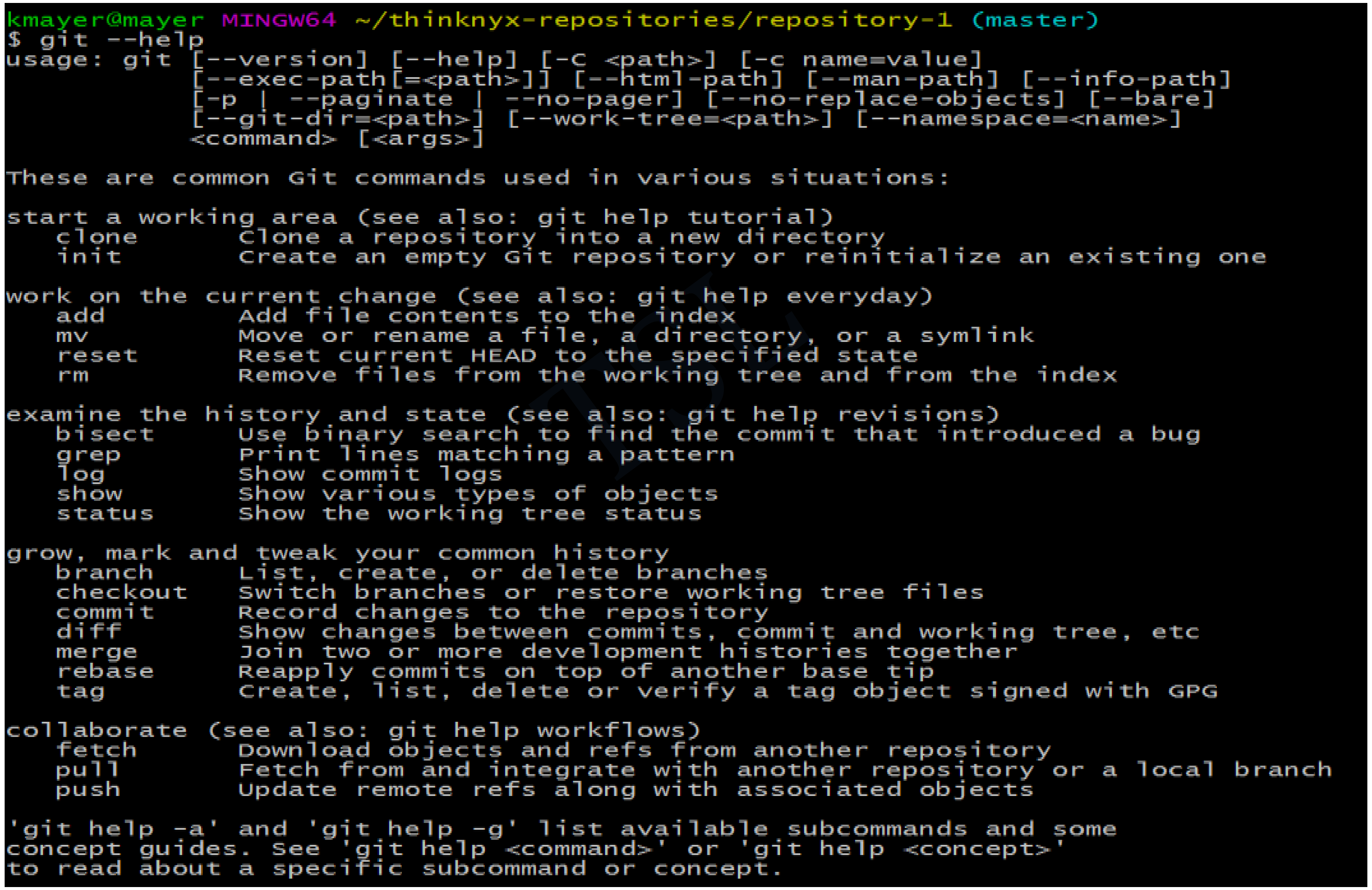
GIT Life Cycle



GIT VERSION

```
[root@user20-master plays]# git --version
git version 1.8.3.1
[root@user20-master plays]#
```

GIT HELP



```
kmayer@mayer MINGW64 ~/thinknyx-repositories/repository-1 (master)
$ git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

Setting Identity in GIT

```
[root@Techlanders ~]# git config --global  user.email
“Gagandeep.singh@Techlanders.com”
[root@Techlanders ~]# git config --global  user.name “Gagandeep Singh”
[root@Techlanders ~]# git config --global -l
user.name=Gagandeep Singh
user.email=Gagandeep.singh@Techlanders.com
[root@Techlanders ~]#
```

GIT Repository

- A **repository** is usually used to organize a single project.
- Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.
- Repository is like a unique shared file system for a project.