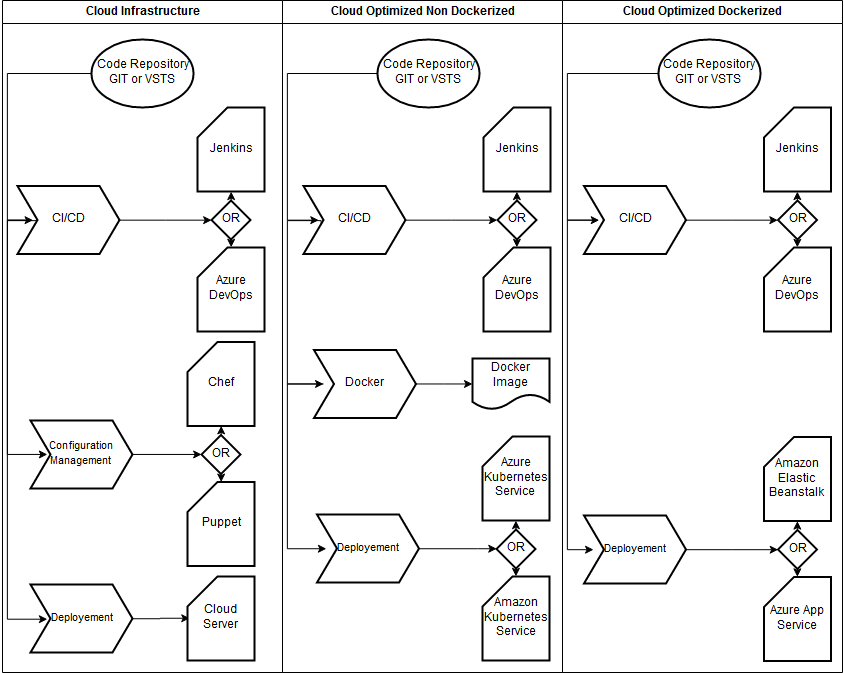
# Cloud Migration Approaches



# Modernize web applications or services and move them to the cloud

* Always not necessary to rearchitect apps into because of cost and time constraints.
* As per the requirement of apps need to ask some questions like:
  + Which apps require a transformation or rearchitecting?
  + Which apps need to be only partially modernized?
  + Which apps can "lift and shift" directly to the cloud?
* We can modernize app with following three types of approaches

1. **Cloud Infrastructure-Ready applications**

* We simply migrate or rehost our current on-premises applications to an infrastructure as a service (IaaS) platform.
* Advantages
  + No rearchitecting, no new code
  + Least effort for quick migration
  + Basic availability guarantees
  + After moving to the cloud, it's easier to modernize even more
* Challenges
  + Smaller cloud value, other than shift in operating expense or closing datacentres
  + Little is managed: No OS or middleware patching; might use infrastructure solutions, like Terraform, Spinnaker, or Puppet

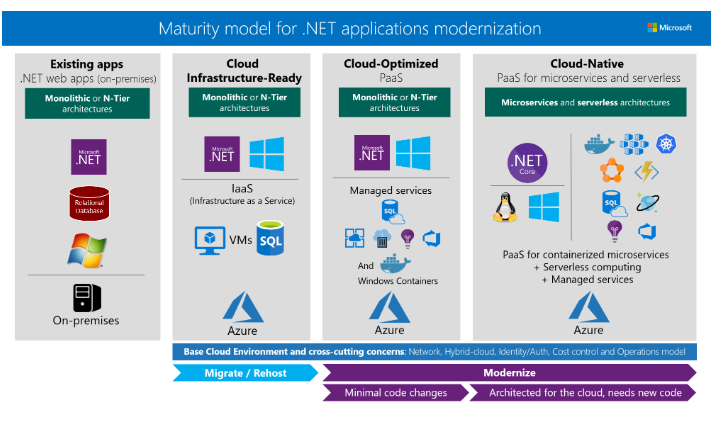
1. **Cloud Optimized applications**

* Without rearchitecting or altering significant code, we can gain additional benefits from running your app in the cloud with modern Technologies like containers and additional cloud-managed services.
* Implement DevOps
* Using technologies like Windows Containers, which is based on Docker Engine.
* We can deploy containers on IaaS or PaaS while using additional cloud-managed services related to databases, cache as a service, Monitoring, and continuous integration/continuous deployment (CI/CD) pipelines.
* Advantages
  + No rearchitecting
  + Minimal code/config changes
  + Improved deployment and DevOps agility to release because of containers
  + Increased density and lower deployment costs
  + Portability of apps and dependencies
  + Flexibility of host targets: PaaS approaches or IaaS
* Challenges
  + Containerizing is an additional step in the learning curve for developers and IT Operations
  + DevOps and CI/CD pipelines is usually ‘a must’ for this approach. If not currently present in the culture of the organization, it might be an additional challenge

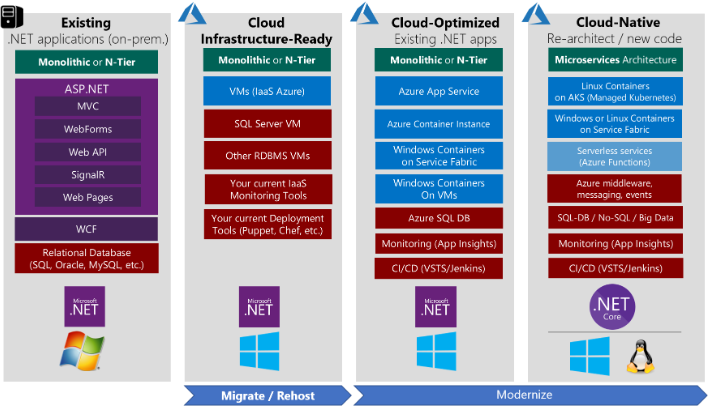
1. **Cloud-Native applications**

* This migration approach typically is driven by business need and targets modernizing your mission-critical applications.
* We use PaaS services to move apps to PaaS computing platforms.
* New code often must be written, especially when we move to cloud-native application and microservice-based models.
* Advantages
  + Architect for the cloud, you get the best benefits from the cloud but new code is needed
  + Micro services cloud-native approaches
  + Modern mission-critical applications, cloud-resilient hyper-scalable
  + Fully managed services
  + Optimized for scale
  + Optimized for autonomous agility by subsystem
  + Built on deployment and DevOps
* Challenges
  + Requires rearchitecture for cloud native apps and micro service architectures and usually requires significant code refactoring or rewriting when modernizing (increased time and budget)
  + DevOps and CI/CD pipelines is usually ‘a must’ for this approach. If not currently present in the culture of the organization, it might be an additional challenge

# Primary technologies for each maturity level for modernizing existing .NET web applications



# Internal technologies that we can use for each maturity level:



# Lift and Shift Existing Monolithic Applications

## Lift and shift existing .NET apps to Azure IaaS (Cloud Infrastructure-Ready)

### **Why migrate existing .NET web applications to Azure IaaS**

* The main reason to migrate to the cloud, even at an initial IaaS level, is to achieve cost reductions. By using more managed infrastructure services, organization can lower its investment in hardware maintenance, server or VM provisioning and deployment, and infrastructure management.

### **When to migrate to IaaS instead of to PaaS**

* If goal is to move app from on premise to Cloud
* The apps that still cannot move to Windows Containers and PaaS such as App Service or orchestrators like Azure Kubernetes Service, migrate those to simple plain VMs (IaaS).

### **For Migration use following tools**

* Azure Migration
* Azure Site Recovery

## Lift and Shift Existing Java Apps to AWS EC2(Cloud Infrastructure-Ready)

### C:\Users\DELL PC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\App-migration-strategies-rehost.jpeg**Why migrate existing Java web applications to Cloud Servers**

* The main reason to migrate to the cloud, much like with .NET applications is to achieve cost reductions. By using more managed infrastructure services, organization can lower its investment in hardware maintenance, server or VM provisioning and deployment, and infrastructure management.

### **When to migrate to IaaS instead of to PaaS**

* If goal is to move app from on premise to Cloud
* If the goal is to shift to a more complex cloud architecture then we can LiftNShift applications to cloud servers since then applications are easier to optimize/re-architect once they’re already running in the cloud.

### **For Migration use following tools**

* AWS CLI for VM import

# Modernize existing monolithic applications to Cloud-Optimized applications

## **Main pillars of a Cloud-Optimized application**

* Containerized
* Resilient and scalable cloud
* Monitoring
* DevOps culture and continuous delivery

## **Microsoft technologies in cloud-optimized application**

* Cloud infrastructure
  + IaaS
* Runtime
  + DockerEngine
    - Run On Linus or Windows container
    - .net application should run windows container
* Managed Cloud
  + When we choose a managed cloud option, we can avoid the expense and complexity of managing and supporting the underlying infrastructure, VMs, OS patches, and networking configuration
  + If we choose to migrate by using IaaS, we are responsible for all of these tasks, and for associated costs.
  + Examples
    - Azure SQL DB
    - Azure Storage
    - Azure redis cache
    - Azure Cosmos DB
    - Azure App Service
    - AKS
    - Etc
* Application Life cycle
  + Azure DevOPs Service
  + Jenkins

# Modernize existing .NET apps to Cloud-Optimized applications

## **Main characteristics of the Cloud-Native model**

* Micro-Services
* Docker Containers
* Orchestrators
  + AKS
  + Service fabrics
* ServerLess
  + Azure Functions
* Code In multiple language

# Pricing

## For .NET Applications

* Azure DevOps is free for open source projects and small projects (up to five users). For larger teams, the cost ranges from $30 per month (10 users) to $6,150 per month (1,000 users).

|  |  |  |
| --- | --- | --- |
| **Service type** | **Region** | **Estimated Cost** |
| App Service | West India | ₹7,237.54 |
| Azure Kubernetes Service (AKS) | West India | ₹11,392.09 |
| Azure DevOps |  | ₹0.00 |
| Storage Accounts | West India | ₹141.71 |
| SQL Database | West India | ₹1,112.97 |
| Support |  | ₹0.00 |
|  |  | Microsoft Online Services Agreement |
|  |  | ₹19,884.31 |

* Azure Costing

## For Java/Open Source Applications

* AWS offers a pay as you go model for billing, and most of AWS services have a per second cost.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technology** | **Required For** | **Pricing** | **Note** | **Monthly Cost estimate** |
| Jenkins(Client on AWS EC2 Instance) | Cloud Optimize, Non Dockerized | Only charge for EC2 Instance: ₹3.03/hr |  | ₹ 2,181.60 |
| Amazon EKS | Cloud Optimized, Dockerized | ₹13.72/hr For EKS | Additional charges for resources used: 1. 16Gb RAM, 4 Core, t3.xlarge,EC2: ₹12.29/hr 2. 20Gb Storage, EBS: ₹7.82 per GB/Month | ₹ 18,883.60 |
| Amazon Elastic Beanstalk | Cloud Optimized, Non Dockerized | No Additional Cost for EB | Only charged for resources used, Ex : 1. 16Gb RAM, 4 Core, t3.xlarge,EC2: ₹12.29/hr 2. 20Gb Storage, EBS: ₹7.82 per GB/Month | ₹ 9,005.20 |