**Network Links & Port Mapping**

|  |  |  |
| --- | --- | --- |
| **Component** | **Port** | **Notes** |
| Jenkins | 8080 | Jenkins web interface |
| Bitbucket | 7990/22 | Git HTTP(S)/SSH |
| SonarQube | 9000 | Static code analysis dashboard |
| Nexus Repo | 8081 | Artifact storage |
| Checkmarx | 443 | Typically web-based dashboard |
| Harbor | 443/4443 | For secure Docker image transfer (HTTPS) |
| Docker Daemon | 2375/2376 | TCP for Docker client/engine |
| Teams Webhook | 443 | Sends notifications via MS Graph |

**ON-PREMIS:**

**Pros:**

|  |  |
| --- | --- |
| **Advantage** | **Description** |
| **Full Control** | You have total control over hardware, data, and configuration. |
| **Security** | Data never leaves your premises – good for strict compliance policies. |
| **Customization** | Can tailor systems/software exactly as per internal needs. |
| **Latency** | Lower network latency within the LAN environment. |

**Cons:**

|  |  |
| --- | --- |
| **Disadvantage** | **Description** |
| **High Initial Cost** | Expensive servers, networking gear, licenses, and setup. |
| **Scalability Issues** | Scaling requires purchasing more hardware – not immediate. |
| **Maintenance Overhead** | You’re responsible for updates, backups, patching, and downtime. |
| **Disaster Recovery Complexity** | Recovery from failure needs in-house planning and offsite backups. |
| **Resource Constraints** | Limited compute/storage during peak loads if not scaled properly. |

**CLOUD:**

**Pros:**

|  |  |
| --- | --- |
| **Advantage** | **Description** |
| **On-Demand Scalability** | Easily scale CI agents, builds, and infra with auto-scaling groups. |
| **Pay-as-You-Go** | Only pay for the compute/network/storage you use. |
| **Managed Services** | Tools like CodeBuild, EKS, RDS, etc., reduce maintenance burden. |
| **High Availability** | AWS offers multi-AZ failover, automated backups, etc. |
| **Global Accessibility** | Teams from anywhere can access and trigger builds securely. |

**Cons:**

|  |  |
| --- | --- |
| **Disadvantage** | **Description** |
| **Recurring Cost** | Ongoing monthly billing, potentially unpredictable. |
| **Data Security Concerns** | Sensitive data needs extra controls like VPC, KMS, IAM, etc. |
| **Vendor Lock-In** | Might get tied to specific AWS services and patterns. |
| **Latency (sometimes)** | Accessing cloud resources can have some network delay. |
| **Learning Curve** | Requires cloud expertise to set up secure, efficient pipelines. |

**MAJOR DIFFERENCE BETWEEN CLOUD AND ONPREMISES**

|  |  |  |
| --- | --- | --- |
| **Feature** | **On-Premises** | **Cloud-Based CI** |
| **Cost** | High upfront | Operational (monthly) |
| **Maintenance** | Manual | Mostly automated |
| **Scalability** | Manual & delayed | Instant & dynamic |
| **Customization** | Full | Limited by provider |
| **Security** | Internal controls | Shared responsibility |
| **Availability** | Depends on infra | High (multi-AZ) |
| **Disaster Recovery** | Manual setup | Automated (Snapshots) |
| **Speed** | Faster internal LAN | Variable (depends) |

**CI Pipeline Setup: On-Premises vs Cloud (AWS)**

**On-Premises CI Pipeline**

1. Provision VMs for Jenkins, SonarQube, Nexus, Checkmarx (if used), and Harbor on a common internal network.
2. Install Java, Jenkins, and required plugins (Git, Pipeline, Docker, etc.).
3. Install and configure SonarQube with PostgreSQL for code quality analysis.
4. Set up Nexus Repository for artifact storage and access via port 8081.
5. Install Docker and deploy Harbor as a private container registry.
6. Set up Checkmarx for static code security scanning.
7. Create Jenkins pipeline stages for build, test, scan, package, and deploy.
8. Integrate Jenkins with SonarQube, Nexus, Harbor, and optionally Checkmarx.
9. Enable access control, TLS/SSL, and configure firewalls for each service.
10. Set up monitoring (Grafana/Prometheus) and backup all critical services regularly.

**Cloud-Based CI Pipeline**

1. Create an EKS (Elastic Kubernetes Service) or EC2-based Jenkins setup in a secure VPC.
2. Use AWS CodeCommit or GitHub for source code versioning.
3. Set up AWS CodeBuild or Jenkins on EC2/EKS for building and testing code.
4. Integrate SonarQube (on EC2 or via SaaS) for code quality analysis.
5. Use AWS CodeArtifact or self-managed Nexus (on EC2) to store build artifacts.
6. Use ECR (Elastic Container Registry) or Harbor (self-hosted) for container image storage.
7. Set up AWS Secrets Manager for managing secrets across your pipeline.
8. Build pipelines using AWS CodePipeline or Jenkins with multi-stage jobs.
9. Ensure IAM roles, security groups, and encryption are properly configured.
10. Use CloudWatch, AWS Backup, and GuardDuty for monitoring, backup, and security.

**Cost Comparison: On-Premises vs Cloud**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **On-Premises** | **Cloud (AWS)** |
| **Initial Cost** | High (hardware, licenses, setup) | Low to moderate (pay-per-use model) |
| **Maintenance Cost** | High (IT staff, updates, power) | Low to moderate (AWS handles infrastructure) |
| **Scalability Cost** | High (manual scaling, hardware limits) | Low (auto-scaling and flexible plans) |
| **Upgrades** | Manual & costly | Automated and included |
| **Monitoring & Security** | Requires tools and staff | Built-in tools (CloudWatch, Guard Duty) |
| **Backup & DR** | Requires custom solutions | Integrated (AWS Backup, S3, etc.) |
| **Operational Cost** | Continuous (electricity, cooling) | Usage-based, optimized per workload |
| **TCO (Total Cost of Ownership)** | High over time | Lower with optimized usage & scaling |