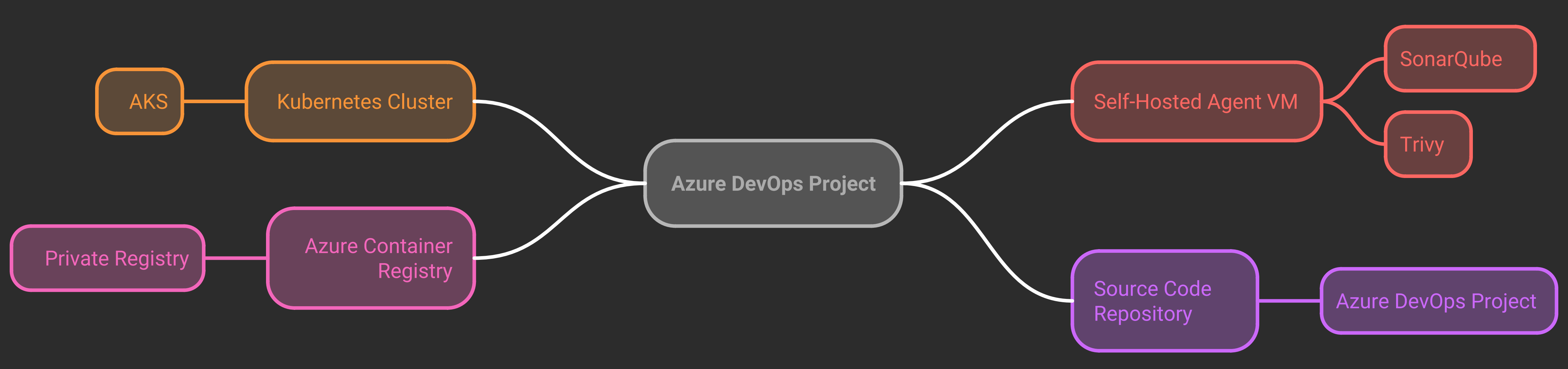
Azure DevOps Project Setup – Java Maven Application

We have implemented a complete **CI/CD pipeline** for a Java Maven project using **Azure DevOps** with the following setup:

Infrastructure & Tools

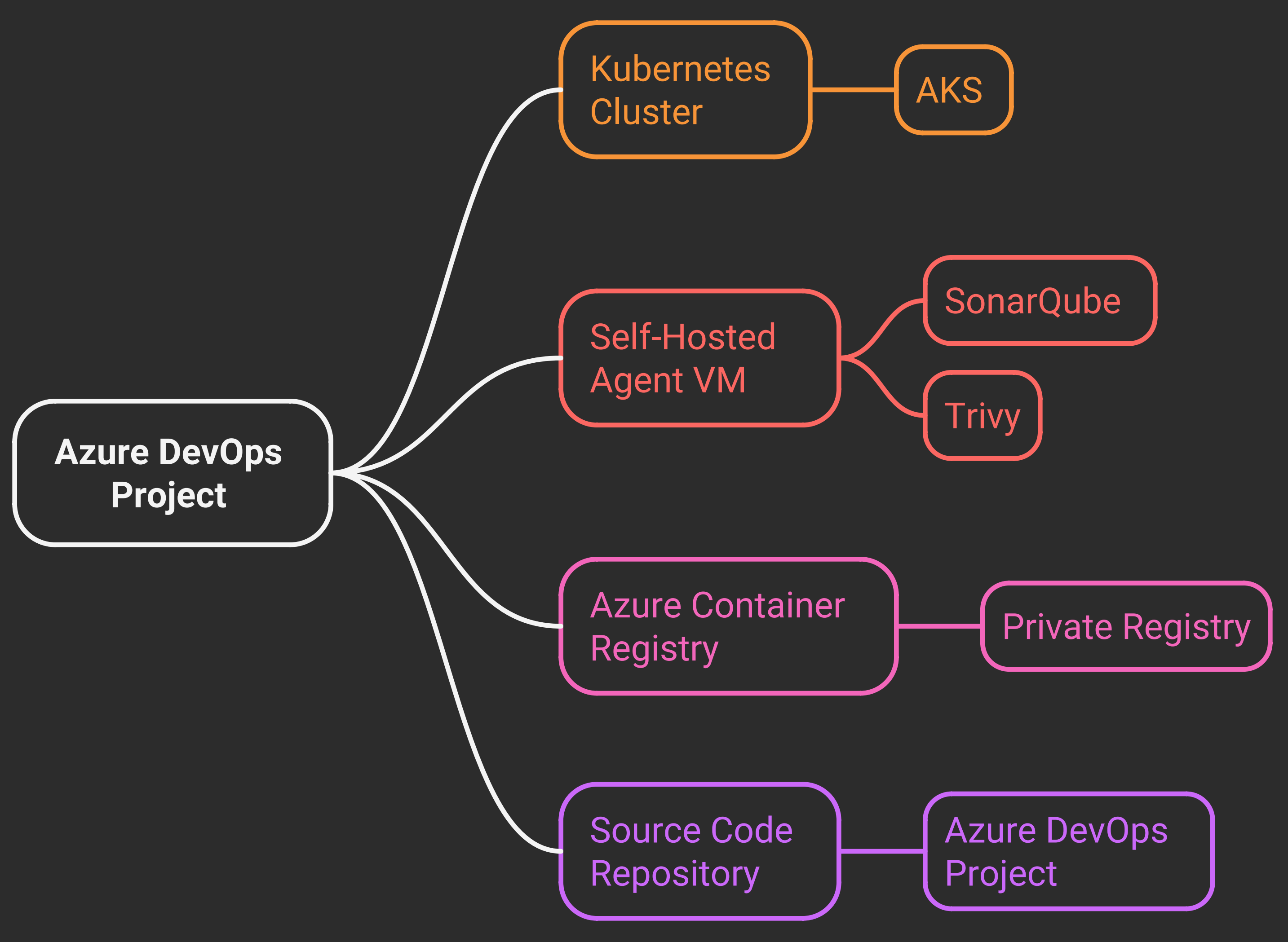
- **Azure DevOps Project** created for source code management and pipeline execution.
- **Kubernetes Cluster** (AKS) provisioned for application deployment.
- **Self-Hosted Agent VM** configured to run Azure Pipelines.
- **Azure Container Registry (ACR)** (private) created for storing Docker images and artifacts.
- **Source Code Repository** maintained in the Azure DevOps project.
- **Agent VM** integrated with the project and configured with additional tools:
    - **SonarQube (containerized)** → for code quality analysis.
    - **Trivy** → for container image security scanning.

Azure DevOps Project Setup for Java Maven Application



Made with Napkin

Azure DevOps Project Setup for Java Maven Application



Made with Napkin

CI Pipeline (Classic Pipeline)

The CI pipeline is configured with multiple tasks/stages:

1. **Run Unit Tests** → Execute test cases for validation.
2. **Copy YAML Files** → Copy Kubernetes manifests from source code repository.
3. **Publish Artifact (drop)** → Store build outputs for further use.
4. **SonarQube Analysis** →
    - Prepare SonarQube analysis.
    - Run code analysis.
    - Publish results.
5. **Maven Packaging** → Build and package the application.
6. **Trivy FS Scan** → Perform filesystem-level security scan.
7. **Docker Build & Push** → Build Docker image and push it to Azure Container Registry.

CD Pipeline (Release Pipeline)

For deployment, a **Release Pipeline** is set up:

- Kubernetes cluster authentication configured.
- YAML manifests are applied to deploy the application on AKS.
- Application is exposed via **External Load Balancer (Public IP)**.
- Deployment verification performed by accessing the application externally.

☑ With this setup:

- CI handles **build, test, quality check, and image scanning**.
- CD handles **deployment and delivery to Kubernetes**.