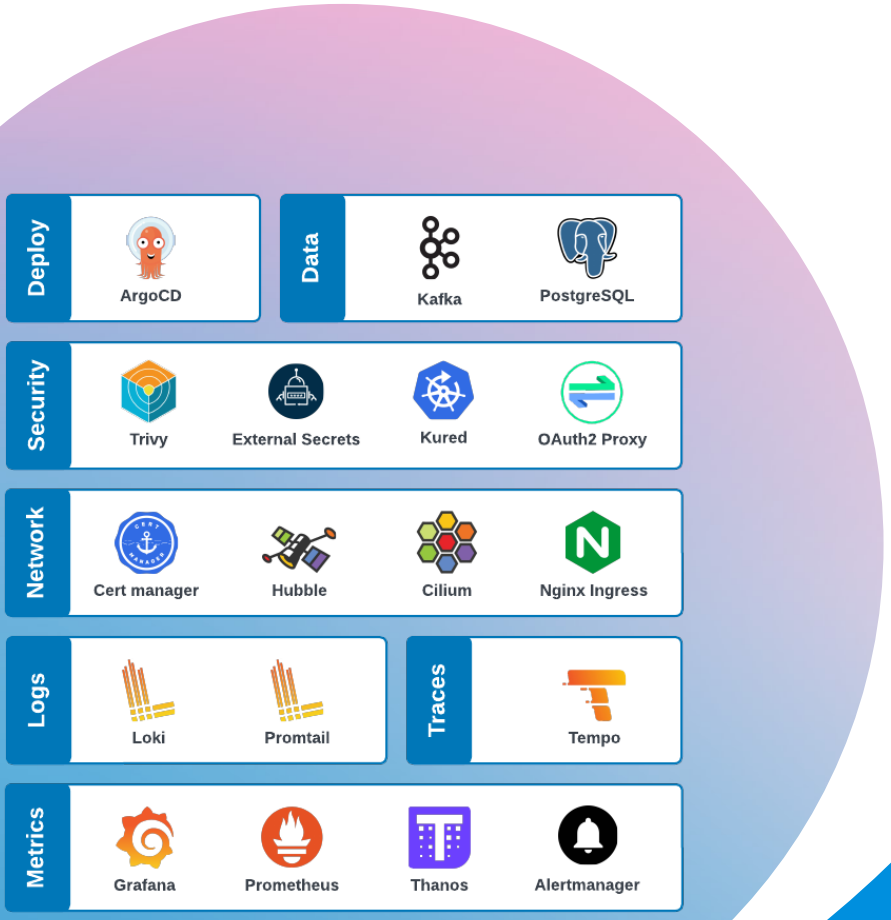


Pionative
Cloud Native Pioneers

Een Kubernetes Referentie Architectuur

Implementeer een productieklaar cluster door deze referentie architectuur te gebruiken op basis van best practices

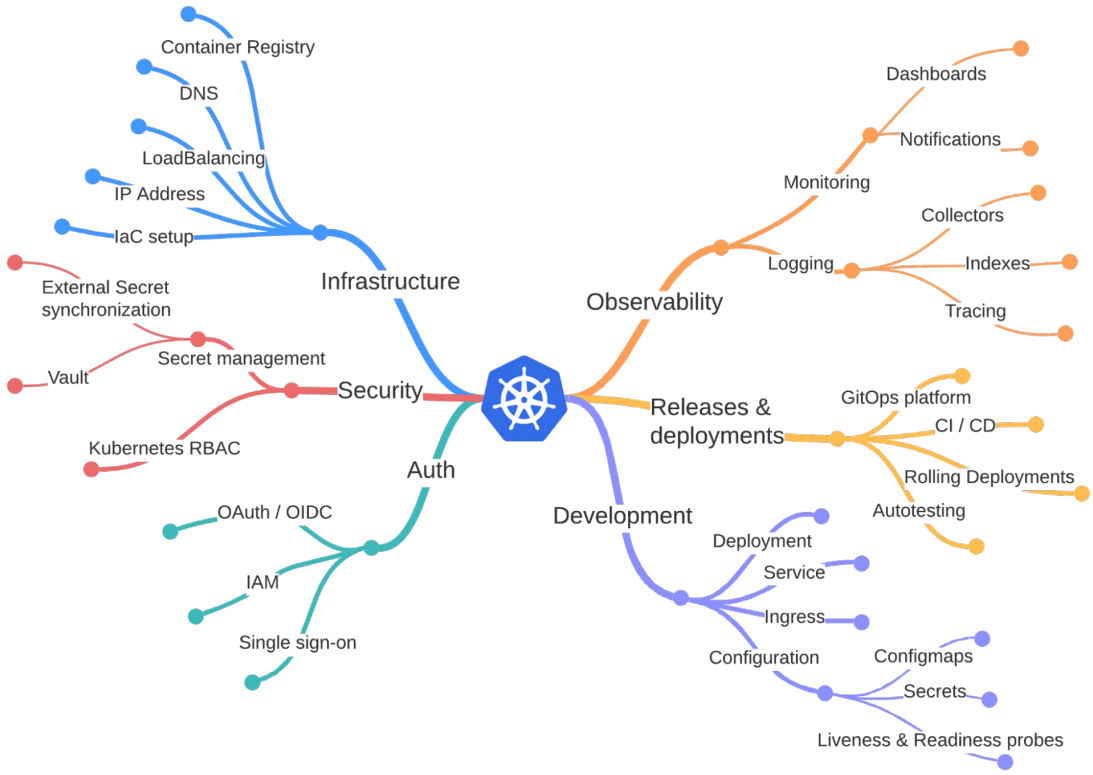
Introductie

Kubernetes is hét platform voor het draaien van containers. Zowel grote als kleine bedrijven gebruiken Kubernetes als standaard om applicaties op te draaien. Maar Kubernetes is erg complex en er zijn veel valkuilen. Eén daarvan is het opnieuw uitvinden van het wiel.

In deze whitepaper laten we zien hoe je Kubernetes succesvol kan implementeren volgens onze referentie-architectuur. We hopen dat dit helpt om de implementatie te versnellen, of om missende delen in de huidige architectuur te identificeren.

Als je meer hulp nodig hebt bij Kubernetes, plan dan eens een introductie meeting. We helpen je graag verder!

2. De uitdaging



Een goed Kubernetes cluster opzetten kost tijd en expertise. Je moet nadenken over alle bovenstaande onderdelen, maar ook over de cloud provider en Managed Services die je wilt gebruiken. Alles moet goed worden opgezet als code en moet het platform continu up-to-date worden gehouden.

Daarnaast zijn Kubernetes platform implementaties vaak erg “opinionated”. Het is lastig marktstandaarden te gebruiken omdat organisaties vaak niet de ervaring hebben met Kubernetes om de juiste keuzes te maken.

Een andere grote uitdaging bij de implementatie van Kubernetes is dat je de focus verliest op de business. Meestal wordt Kubernetes geïmplementeerd door de Lead Engineer. Een expert inhuren is erg duur, en Kubernetes engineers vinden vaak het wiel opnieuw uit, wat erg veel tijd kost.

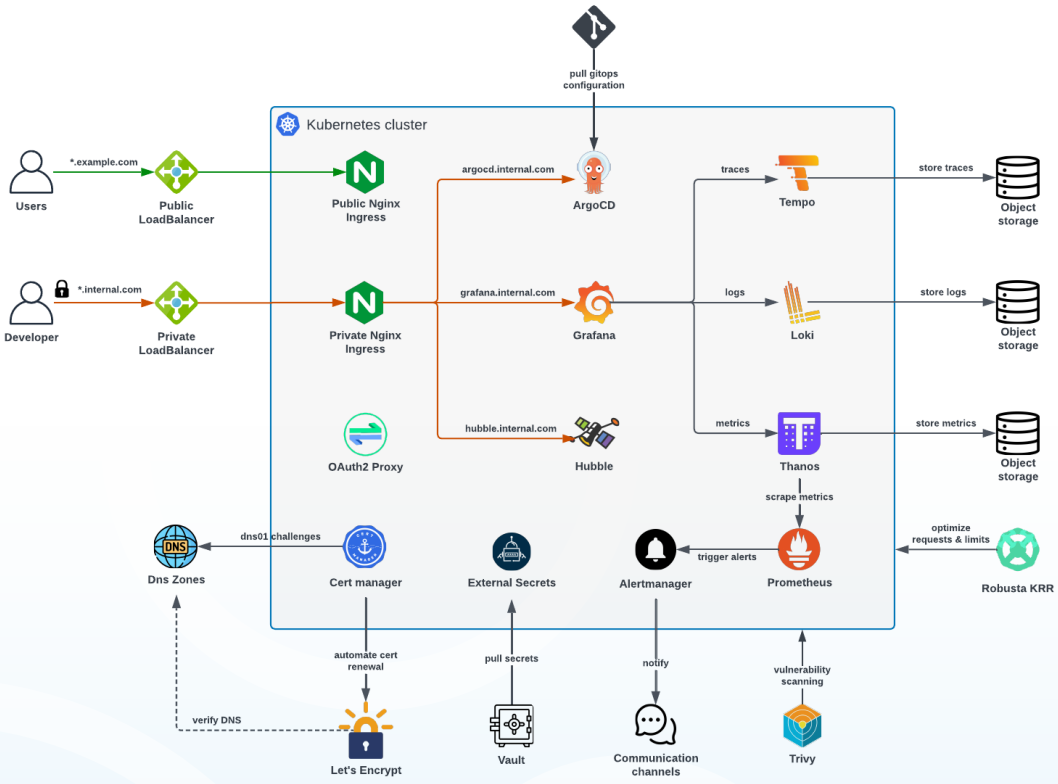
3. Onze referentiearchitectuur

Deze referentie-architectuur helpt om een productieklaar product zo snel mogelijk neer te zetten. Alle componenten in de architectuur zijn vervangbaar. Dat betekent dat er geen technologie-lock ontstaat.

Daarnaast dient deze referentie-architectuur als een stabiele basis. Dat betekent dat voor sommige bedrijven extra componenten nodig zijn en dat deze componenten kunnen worden gebruikt als basis voor de meeste bedrijven.

Bij Pionative hebben we tientallen productie-clusters gebouwd en in beheer op basis van deze referentie-architectuur. Deze technologieën bewijzen zichzelf dagelijks in productie. Kijk eens op onze website wat onze klanten hierover zeggen.

Vind je deze referentie-architectuur interessant? Plan dan eens een gratis architectuur-review sessie om jullie huidige stack te laten reviewen en missende onderdelen in de stack te ontdekken!



3.1. De keuze voor een Kubernetes Distributie

Kubernetes is platform onafhankelijk. Dit betekent dat het vrij gemakkelijk is om applicaties te migreren tussen cloud providers, of zelfs te migreren naar een on-premise of private cloud.

Managed Kubernetes Services

Bij het kiezen van een Kubernetes oplossing heeft een Managed Kubernetes Service altijd de voorkeur. Dit houdt in dat de cloud provider het Kubernetes beheer voor zijn rekening neemt. Complexe onderdelen van het Kubernetes patform, zoals de API-server of etcd worden dan door de provider beheerd. Dit scheelt niet alleen in opstartkosten, maar maakt het beheer ook een stuk gemakkelijker.

Bij Pionative kiezen we vaak voor een Managed Kubernetes Service in Azure (AKS), AWS (EKS) of Google Cloud (GKE). Dit zijn de 3 grootste cloud providers die met elkaar concurreren op prijs en functionaliteit. Al deze cloud providers hebben een stabiele Managed Kubernetes Service en bieden daarnaast Object Storage, Databases en andere managed services.

Self-managed Kubernetes

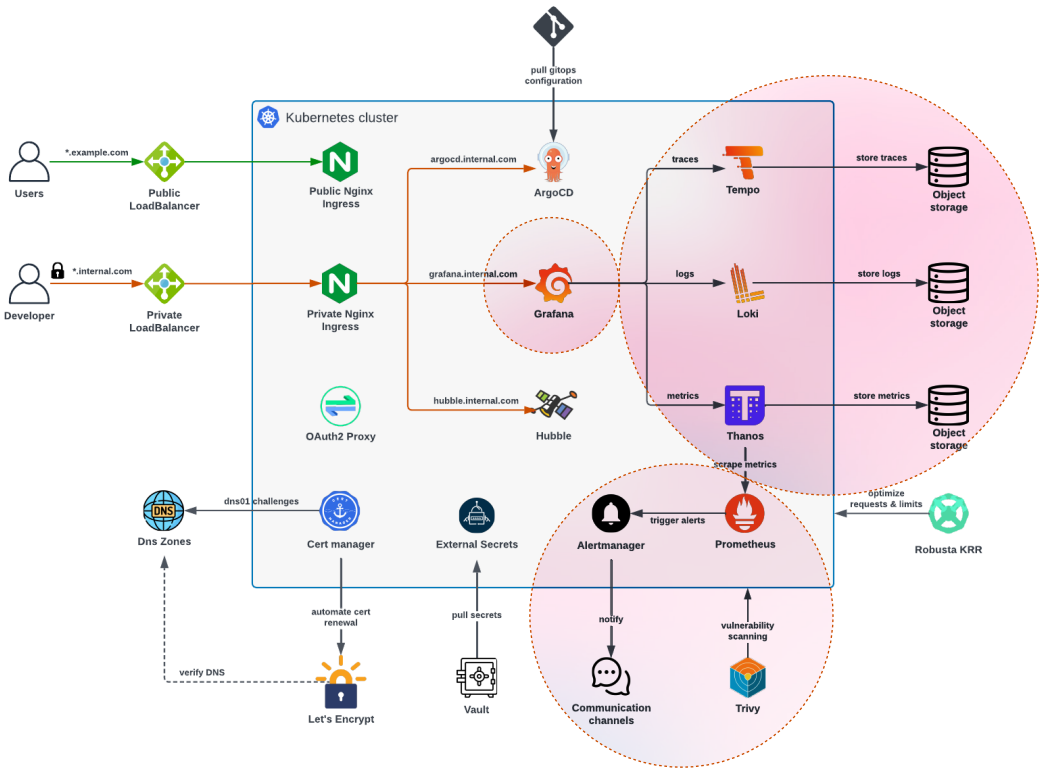
Voor sommige bedrijven kan een on-premise omgeving of private cloud omgeving interessant zijn. Soms wordt vanwege kosten, security of andere redenen gekozen voor een self-managed oplossing. Houd er in dit geval rekening mee dat de implementatie- en beheerkosten hoger zullen liggen.

Als de keus wordt gemaakt om niet voor een Managed Kubernetes Service te gaan, raden we bij Pionative aan om Rancher RKE2 te gebruiken. Dit is een Kubernetes distributie van Rancher, die Open Source beschikbaar is. Daarnaast is deze distributie enorm stabiel en kan het gemakkelijk worden geïnstalleerd met behulp van Infrastructure as Code (IaC).

Best practices:

- Gebruik bij voorkeur een Managed Kubernetes Service
- Gebruik Infrastructure as Code bij de opzet van cloud resources
- Draai Kubernetes in een Private Netwerk

3.2. Observability stack opzetten



Voordat een Kubernetes cluster klaar is voor productie moet een observability stack worden opgezet. Zo kunnen problemen preventief worden opgelost en kan downtime worden voorkomen. Een goede observability stack bestaat uit 3 centrale componenten: Metrics, Logs en Traces. De combinatie hiervan, met name Metrics en Logs, geven extra zichtbaarheid in de infrastructuur en applicaties.

Metrics

Voor het vergaren van metrics gebruiken we de Prometheus Stack. In Kubernetes zal de Prometheus Stack automatisch metrics van alle applicaties ophalen in het cluster. Deze metrics worden opgeslagen in de Prometheus database.

Grafana wordt met de Prometheus Stack uitgerold om zichtbaarheid te geven in de monitoring met behulp van dashboards. En als er iets mis gaat in de stack, bijvoorbeeld bij een fout in de applicatie, zal de Alertmanager een notificatie sturen naar een Teams of Slack kanaal. Zo kunnen problemen snel worden opgelost.

Voor de lange termijn opslag van metrics (bijvoorbeeld maanden of jaren) gebruiken we Thanos. Dit component heeft een connectie met een externe Object Storage. Zo worden alle metrics buiten het Kubernetes cluster opgeslagen en kunnen analyses worden losgelaten op bepaalde patronen over langere tijd.

Logs

Een andere belangrijk component is Centrale Logging. In onze referentie-architectuur gebruiken wij Grafana Loki en Promtail. Door middel van deze componenten kan alle logging in het cluster geautomatiseerd op een centrale plek worden opgeslagen.

Grafana Loki kan worden gebruikt in de Grafana applicatie, en zo kunnen, net zoals bij ElasticSearch of Kibana, de logs van het cluster worden opgevraagd door middel van zoekcriteria. Wij zien een enorm goede performance bij Loki en raden dit dus ook aan bij de meeste van onze klanten.

Ook voor Grafana Loki slaan we lange termijn logs op buiten het cluster in Object Storage. Afhankelijk van de cloud-omgeving kiezen we de meest praktische Object Storage oplossing.

Traces

Een meer geavanceerd component in observability is distributed tracing. Als je een microservices-architectuur hebt kunnen traces goed helpen om performance bottlenecks op te sporen. Daarnaast geven traces een totaaloverzicht van het verkeer dat op het cluster plaatsvindt.

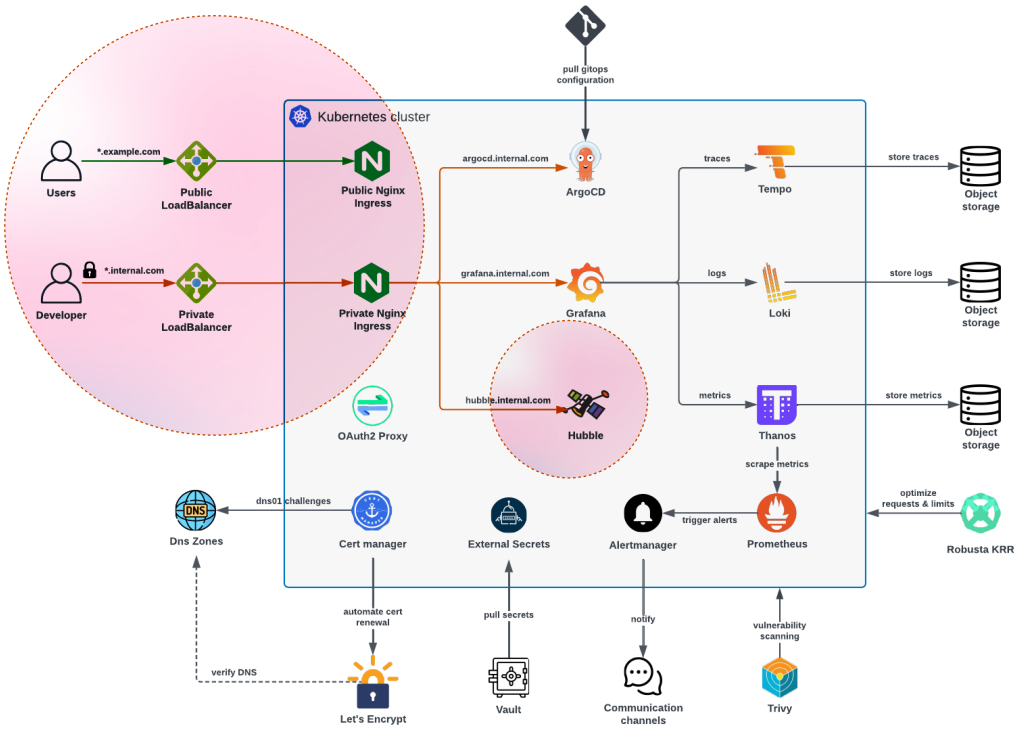
Er zijn verschillende oplossingen voor traces in Kubernetes. Zowel Jaeger als Grafana Tempo zijn erg goed in het visualiseren van traces. Wij gebruiken Grafana Tempo omdat de Grafana User Interface al bekend is bij developers vanwege de Grafana Dashboards en Grafana Loki.

Zowel Metrics als Logs worden in onze referentie-architectuur automatisch opgeslagen. We raden altijd aan om per platform component (bijvoorbeeld Nginx) te kijken naar dashboards en alerts die gemaakt zijn voor dit component. Zo wordt de Kubernetes stack volwassen en kunnen problemen van componenten worden voorkomen.

Best practices:

- Voeg custom alerts and dashboards toe
- Gebruik gestructureerde logging voor betere leesbaarheid
- Sla de logs en metrics op voor de lange termijn

3.3. Netwerk and Load Balancing



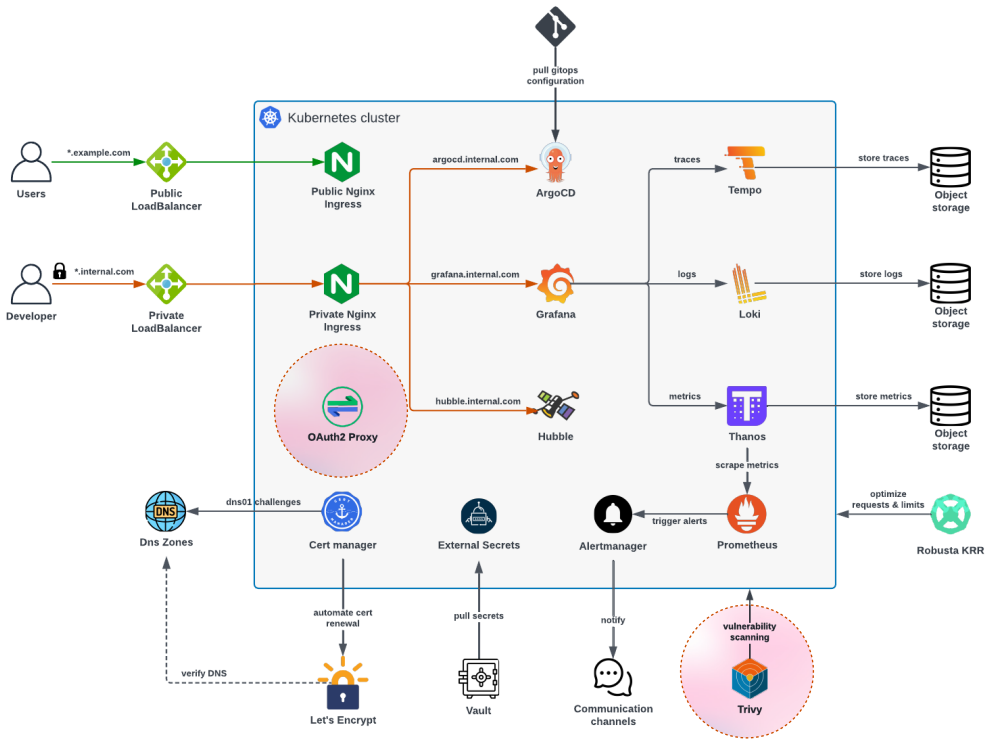
Wij implementeren eigenlijk altijd 2 Load Balancers in elke Kubernetes stack om het publieke en interne verkeer te scheiden. Applicaties die op internet beschikbaar zijn kunnen door iedereen worden aangesproken, maar interne applicaties kunnen alleen worden benaderd over VPN. Dit beveiligt het cluster en geeft meer controle.

In onze referentie-architectuur gebruiken we Nginx als de Ingress Controller. De reden hiervoor is dat Nginx de meestgebruikte Ingress Controller voor Kubernetes is. Het is daarmee erg stabiel, en configuratie is erg gemakkelijk. Ingress is één van de belangrijkste componenten in de stack omdat het kan worden gezien als Single-Point-Of-Failure. Al het verkeer stroomt hier namelijk doorheen.

Best practices:

- Implementeer specifieke alerts en dashboards voor Nginx
- Draai de Ingress Controller als Daemonset in productie

3.4. IAM and Beveiliging



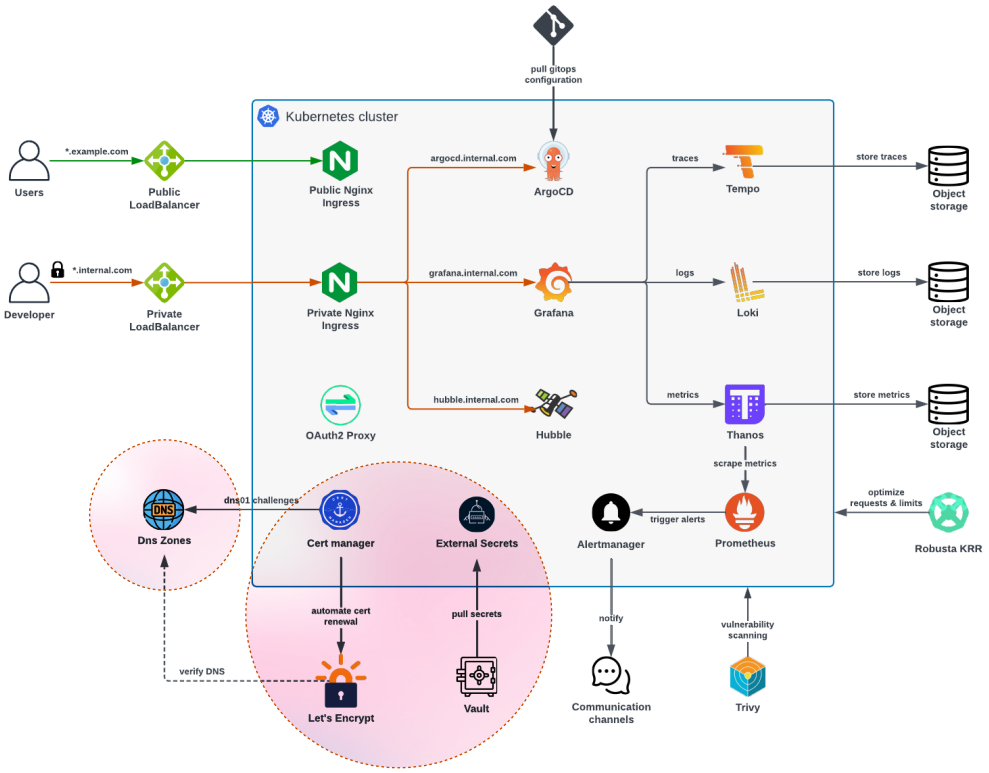
Door het “Principle of Least Privilege” te gebruiken krijgen developers alleen de toegang die ze nodig hebben. Daarom is RBAC en IAM zo belangrijk. We gebruiken bij elk component dus ook een OIDC Login met een centrale Identity Provider zoals Azure Entra ID. Voor componenten die dit niet ondersteunen (zoals Prometheus of Hubble) gebruiken we OAuth2-Proxy om ontwikkelaars te authenticeren.

Daarnaast is het belangrijk om een security scanner te installeren. Wij gebruiken Trivy op Kubernetes om automatisch nieuwe kwetsbaarheden te ontdekken. Als een nieuwe kwetsbaarheid is ontdekt kan bijvoorbeeld een notificatie worden gestuurd naar een Teams of Slack channel om snel een mitigatie te kunnen implementeren.

Best practices:

- Gebruik een betrouwbare Identity Provider zoals Microsoft of Google
- Patch alle componenten regelmatig om kwetsbaarheden te beperken

3.5. Wachtwoorden & Certificaten



Elk Kubernetes cluster heeft zogenaamde “secrets”. Bijvoorbeeld een Database wachtwoord of API key die via een omgevingsvariabele wordt ingeladen. Deze secrets moeten veilig worden opgeslagen en alleen geautoriseerde ontwikkelaars zouden toegang moeten hebben tot deze secrets. Daarom gebruiken wij vaak Secret Managers van cloud providers om dit in op te slaan.

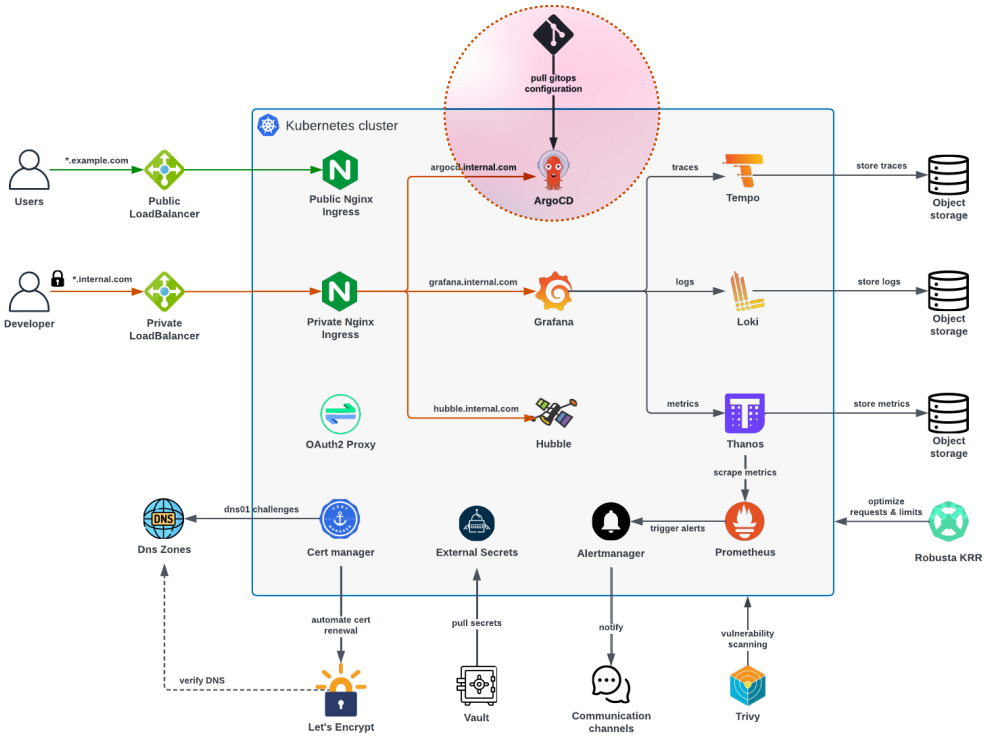
Het Kubernetes platform kan het secret uit de Secret Manager ophalen door middel van de External Secrets Operator. Zo hoeft een ontwikkelaar alleen een referentie naar het secret toe te voegen in Kubernetes. De daadwerkelijke waarde blijft daarmee verborgen.

Voor HTTPS SSL Certificaten raden we aan om LetsEncrypt te gebruiken. Hiermee kunnen automatisch certificaten worden verlengd, zonder dat een handmatige actie nodig is. In sommige sectoren (bijvoorbeeld de financiële sector) kan LetsEncrypt niet worden gebruikt. In dat geval slaan we de SSL Certificaten handmatig op in een Secret Manager om ze vervolgens met External Secrets Operator weer in het cluster op te slaan.

Best practices:

- Gebruik zoveel mogelijk Managed Identity in de cloud omgeving om het gebruik van API keys te beperken
- Scheid de secret oplossing van de applicatielogica
- Blokkeer de toegang tot secrets in productie voor de meeste engineers

3.6. Kubernetes Deployments



De nieuwste standaard voor het deployen op Kubernetes is GitOps. Met GitOps wordt de volledige "deployment state" opgeslagen in GIT. Dit maakt het gemakkelijker om wijzigingen uit te rollen, terug te draaien, of te zien wat er in het verleden is gebeurd.

De twee meest gebruikte GitOps Operators zijn FluxCD en ArgoCD. In onze referentie-architectuur gebruiken we ArgoCD, aangezien de developer-experience zo goed is. ArgoCD heeft een fantastisch dashboard en helpt ontwikkelaars om sneller te werken.

GitOps werkt met alle GIT providers. Het maakt dus niet uit of Azure DevOps GIT, Bitbucket, Github, Gitlab of een andere Git provider wordt gebruikt.

Best practices:

- Gebruik het app-of-apps pattern in ArgoCD om alle applicaties gemakkelijk uit te rollen
- Zet automatische synchronisatie van ArgoCD aan
- Gebruik een aparte GitOps Git repository voor de Kubernetes configuratie

4. Volgende stap

Nadat we door de referentie-architectuur hebben gelopen hopen we dat je een goede basis hebt om te starten met Kubernetes. Of als je al een bestaand Kubernetes cluster hebt, hopen we dat dit helpt om missende onderdelen te implementeren.

Zelf verder implementeren

Ben je geïnspireerd door deze referentie-architectuur en wil je meer weten? We zijn al een tijd bezig met een complete Kubernetes Guide die een stuk dieper gaat dan deze whitepaper. Ben je hierin geïnteresseerd? Stuur ons dan een berichtje, dan sturen we deze graag op zodra het beschikbaar is!

Kun je nog hulp gebruiken?

Natuurlijk is dit nog maar het tipje van de sluier van wat de Pionative Architectuur te bieden heeft. Wil je eens samen met ons door jullie architectuur heen lopen, of heb je vragen waarom we bepaalde keuzes hebben gemaakt? Dan helpen we je graag door middel van een gratis architectuur sessie.

Als je snel en voordelig naar productie wilt met een Kubernetes omgeving, hebben wij **deze volledige architectuur als code voor je klaarliggen**. We helpen je graag om een productieklare architectuur neer te zetten in **jouw cloud omgeving**. Dit duurt normaal gesproken een week, en **we geven je ook nog alle code**!

Wil je meer weten over Pionative, of wil je meer weten over wat onze klanten over ons zetten, kijk dan eens op onze website!

Pionative
Cloud Native Pioneers

Benieuwd hoe deze Architectuur jouw organisatie verder kan helpen?

Plan een gratis consultatie

Pionative B.V.
Eendrachtlaan 102
3526 LB Utrecht
Nederland

info@pionative.com
+31 6 211 987 53