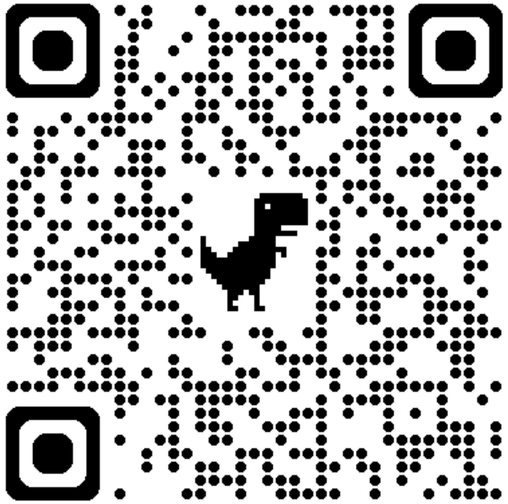
Architectural Patterns Focus: Security

Daniel Hinojosa

Design Pattern Security Standards

In this Presentation

- Types and History of Attacks
- Threat Modeling
- Gateway Pattern
- Sidecar/Ambassador Pattern
- mTLS
- OWASP
- Supply Chains
- More...



Slides and Material: https://github.com/dhinojosa/nfjs-architectural-patterns-security

Types of Cyberattacks

Phishing Attacks

- Attempt to deceive individuals into providing sensitive information
- Typically carried out through email, instant messaging, credit card numbers
- To protect against phishing attacks:
  - Individuals and organizations should be aware of the signs of phishing attempts, such as unsolicited requests for sensitive information, suspicious links or attachments, and messages with poor grammar or unusual language.
  - Implementing security measures like multi-factor authentication, regular security training, and using advanced email filtering technologies can also help mitigate the risk of phishing attacks.

Voice-Phishing Attacks

- Cyber attack where attackers use phone calls to deceive individuals into providing sensitive information.
- Vishing combines “voice” and “phishing” and relies on social engineering techniques to manipulate victims.
- To protect against vishing attacks:
  - Be cautious when receiving unsolicited phone calls requesting personal information
  - Verifying the caller’s identity by contacting the organization directly using a known phone number
  - Education of employees

Smishing Attacks

- A smishing attack, or SMS phishing, is a type of cyber attack where attackers use text messages to deceive individuals into providing sensitive information or performing actions that compromise their security.
- “SMS” (Short Message Service) and “phishing” and relies on social engineering techniques to manipulate victims.
- To protect against smishing attacks:
  - Be cautious when receiving unsolicited text messages that request personal information or contain links
  - Verifying the legitimacy of the message by contacting the organization directly using known contact information, avoiding clicking on links in suspicious messages, and being aware of common smishing tactics can help mitigate the risk.
  - ecurity features like multi-factor authentication and keeping mobile devices updated with the latest security patches

Pharming Attacks

- A pharming [spoofing] attack is a type of cyber attack where an attacker redirects the traffic from a legitimate website to a fraudulent one without the user's knowledge.

- Pharming is particularly dangerous because it can be difficult for users to detect, even if they are vigilant about checking URLs and other security indicators.

- To protect against pharming attacks:

  - Employ DNS services that offer protection against DNS poisoning (DNS cache poisoning or DNS spoofing, is a type of cyber attack where an attacker corrupts the Domain Name System (DNS) cache, causing the DNS server to return incorrect IP addresses)

  - Keep Operating Systems Updated

Non-Payment/Non-Delivery Attacks

- A non-payment attack is a type of scam where an attacker promises goods or services but fails to pay for them.
- This type of attack can occur in various contexts, such as online marketplaces, freelance work, or business transactions
- To protect against non-payment attacks:
  - Verify buyers - Ensure identity and reputation
  - Use Secure Payments - Use secure and reliable methods
  - Check Payment Validity - Ensure payment is made before delivery
  - Set Clear Terms and Conditions

Extortion Attacks

- Cyber attack where an attacker threatens to cause harm or inflict damage unless the victim meets specific demands, usually involving the payment of money.
- These attacks can take various forms and leverage different types of threats to coerce the victim: ransoms, coercion, DOS attacks, blackmail
- To protect against extortion attacks:
  - Regular Backups
  - Cybersecurity Measures
  - Rate Limiting
  - Legal and Law Enforcement

Notable Cyberattack Events

Target Missed Signs of a Data Breach

Share full article

By Elizabeth A. Harris and Nicole Perlroth

March 13, 2014

Target acknowledged on Thursday that its computer security system had alerted it to suspicious activity after hackers infiltrated its network last year, but the company ultimately decided to ignore it, allowing what would become one of the largest data breaches ever recorded to proceed without a hitch.

"With the benefit of hindsight, we are investigating whether, if different judgments had been made, the outcome may have been different," Molly Snyder, a spokeswoman for Target, said in a

Target Attack of 2013

- **Attack Type**: Phishing Attack
- **Damage**: 40 million credit and debit cards of shoppers who had visited its stores during the 2013 holiday season
- **Result**: Paid $18.5 million to settle claims. Forced Credit card companies to move to chip cards

ATM Fraud Linked In RBS WorldPay Card Breach

Thieves Net $9 Million in 30 Minutes

Linda McGlasson • February 5, 2009

In what is being called a well-orchestrated ATM card scam, the true extent of RBS WorldPay's public announcement in late December that its computer systems had been hacked in November has been revealed. In a news report on Wednesday, FBI law enforcement said that a network of thieves withdrew $9 million from 130 ATMs in 49 cities around the world just after midnight on November 8 with cloned cards created from stolen data taken in the RBS WorldPay hack.

RBS
The Royal Bank of Scotland

Royal Bank of Scotland of 2008

- **Attack Type**: The attackers were able to gain access to a database containing the account numbers and personal identification numbers (PINs) of payroll debit cards that the company's customers give to their employees in lieu of live paychecks or direct deposits
- **Damage**: 1.5 million cardholders money stolen totaling $9 million in 30 minutes
- **Result**: Head hacker was convicted. Worth noting that RBS has had other incidents and vulnerability reports since.

ALERT

Reported Supply Chain Compromise Affecting XZ Utils Data Compression Library, CVE-2024-3094

Release Date: March 29, 2024

CISA and the open source community are responding to reports of malicious code being embedded in XZ Utils versions 5.6.0 and 5.6.1. This activity was assigned CVE-2024-3094. XZ Utils is data compression software and may be present in Linux distributions. The malicious code may allow unauthorized access to affected systems.

CISA recommends developers and users to downgrade XZ Utils to an uncompromised version—such as XZ Utils 5.4.6 Stable—hunt for any malicious activity and report any positive findings to CISA.

See the following advisory for more information:

- Red Hat: Urgent security alert for Fedora 41 and Rawhide users

XZ Utils Attack

- **Attack Type**: Multi-year social engineering; supply-chain attack. Installed a backdoor
- **Damage**: They were likely very close to seeing the backdoor update merged into Debian and Red Hat
- **Result**: Jia Tan, the pseudonym of the perpetrator, is not found.

BLOG

Apache Log4j Vulnerability Guidance

Released: April 08, 2022

RELATED TOPICS: CYBER THREATS AND ADVISORIES

Summary

Note: *CISA will continue to update this webpage as well as our community-sourced GitHub repository as we have further guidance to impart and additional vendor information to provide.*

CISA and its partners, through the Joint Cyber Defense Collaborative, are responding to active, widespread exploitation of a critical remote code execution (RCE) vulnerability (CVE-2021-44228) in Apache's Log4j software library, versions 2.0-beta9 to 2.14.1, known as "Log4Shell." Log4j is very broadly used in a variety of consumer and enterprise services, websites, and applications—as well as in operational technology products—to log security and performance information. An unauthenticated remote actor could exploit this vulnerability to take control of an affected system.

(Updated April 8, 2022) Organizations should continue identifying and remediating vulnerable Log4j instances within their environments and

Log4Shell

- **Attack Type**: Vulnerability exploit.
- **Damage**: Log4Shell, also known as the Log4j vulnerability, is a remote code execution (RCE) vulnerability in some versions of the Apache Log4j 2 Java library
- **Result**: Used in 2022 by Chinese Government to attack U.S. State websites. Used in 2021 to attack the Belgium Ministry of Defence. Used in 2021 by varying countries. It remains popular and continues to this day.

2.5 Million More People Potentially Exposed in Equifax Breach

Share full article

Equifax has been reeling since its announcement last month that hackers exploited its website to extract sensitive personal information on potentially millions of consumers. Tami Chappell/Reuters

EQUIFAX®

Equifax Cyberattack

- **Attack Type**: Vulnerability exploit.
- **Damage**: Private records of 147.9 million Americans along with 15.2 million British citizens and about 19,000 Canadian citizens were compromised in the breach, making it one of the largest cybercrimes related to identity theft
- **Result**: The United States Department of Justice announced on February 10, 2020 that they had indicted four members of China's military on nine charges related to the hack, though there has been no additional evidence that China has since used the data from the hack. Equifax settled with the Federal Trade Commission and the Financial Conduct Authority

Threat Modeling

Threat Modeling

- Used to think about security requirements early in the development cycle
- Aims to highlight the risks in an ecosystem as a simple matrix with the likelihood and impact of the risk and a corresponding risk mitigation strategy if it exists

Threat Modeling Goals

**After a successful threat modeling session,
you should be able to define the following:**

- **Asset**: A property of an ecosystem that you need to protect
- **Security control**: A property of a system that protects the asset
- **Threat actor**: An entity or organization who exploit risks
- **Attack surface**: The part of the system that the threat actor is interacting with
- **Threat**: The risk to the asset
- **Mitigation**: Defines how to reduce the likelihood and impact of a threat

Typical Threat Actors

- **End User**: Can connect to the application, entry point is usually the load balancer or ingress
- **Internal Attacker:** Limited Access in your cluster or cloud
- **Privileged Attacker:** Administrator type access

What does a Threat Model Look Like?

Assets	Threat	Security Control	Mitigation
kube-apiserver	Weak authentication because of the use of self-signed certificates	Enable client CA using `--client-ca-file`	Monitor ingress connections to kube-apiserver to check for anomalies
etcd	Data is not encrypted at rest by default	Encrypt data using `--encryption-provider-config`	Pass the configuration parameter `--encryption-provider-config` to kube-apiserver by default
kubelet	kubelet endpoints (unauthenticated) can be used to compromise nodes and containers.	Authentication / authorization	Provide the CA bundle with kubelet to ensure authentication
Pods	Pods can run as root users and compromise the host	PodSecurityAdmission	Rarely do pods need root privileges in the regular workflow. Pod Security Admission can ensure pods are not run as root.

Gateway Pattern

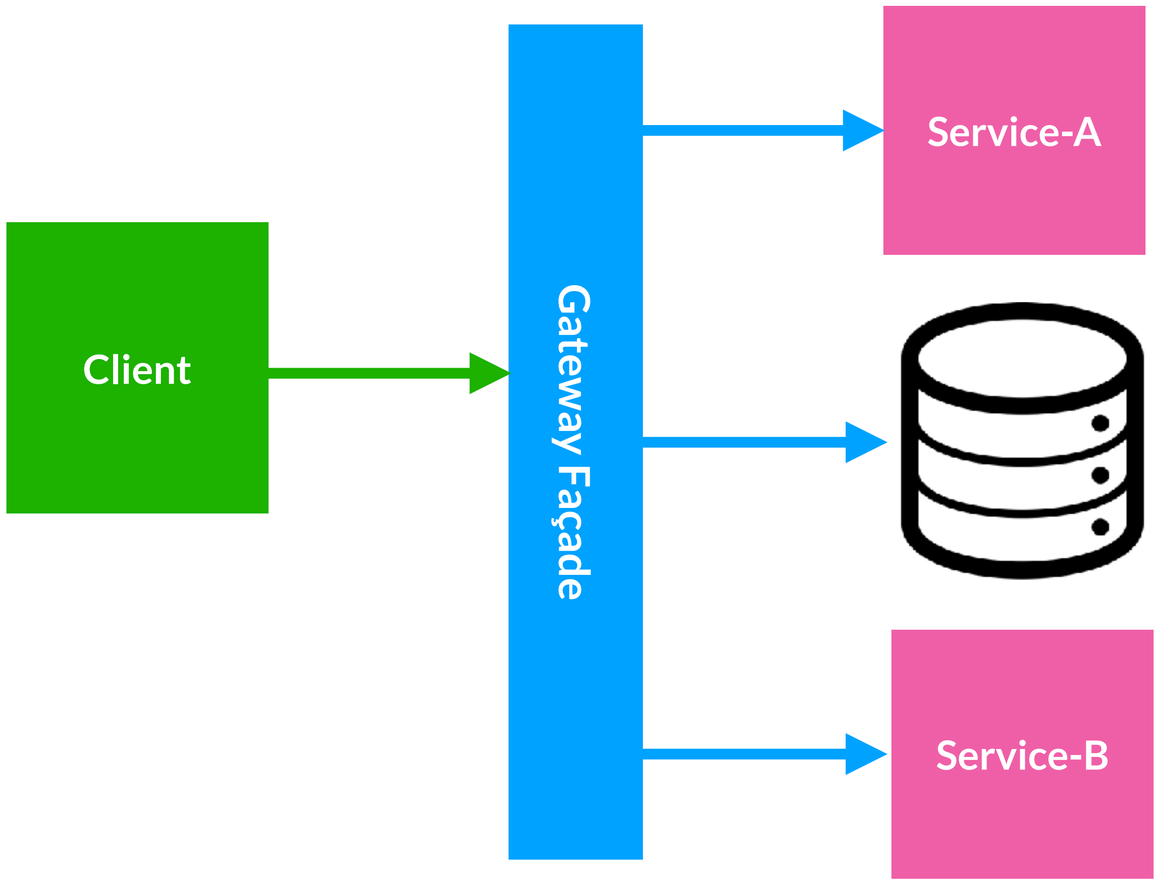
The Problem

- Services may have potentially sensitive information
- If a hacker is able to get into the hosting environment, they can have access to a gamut of information like credentials, storage keys and datastores

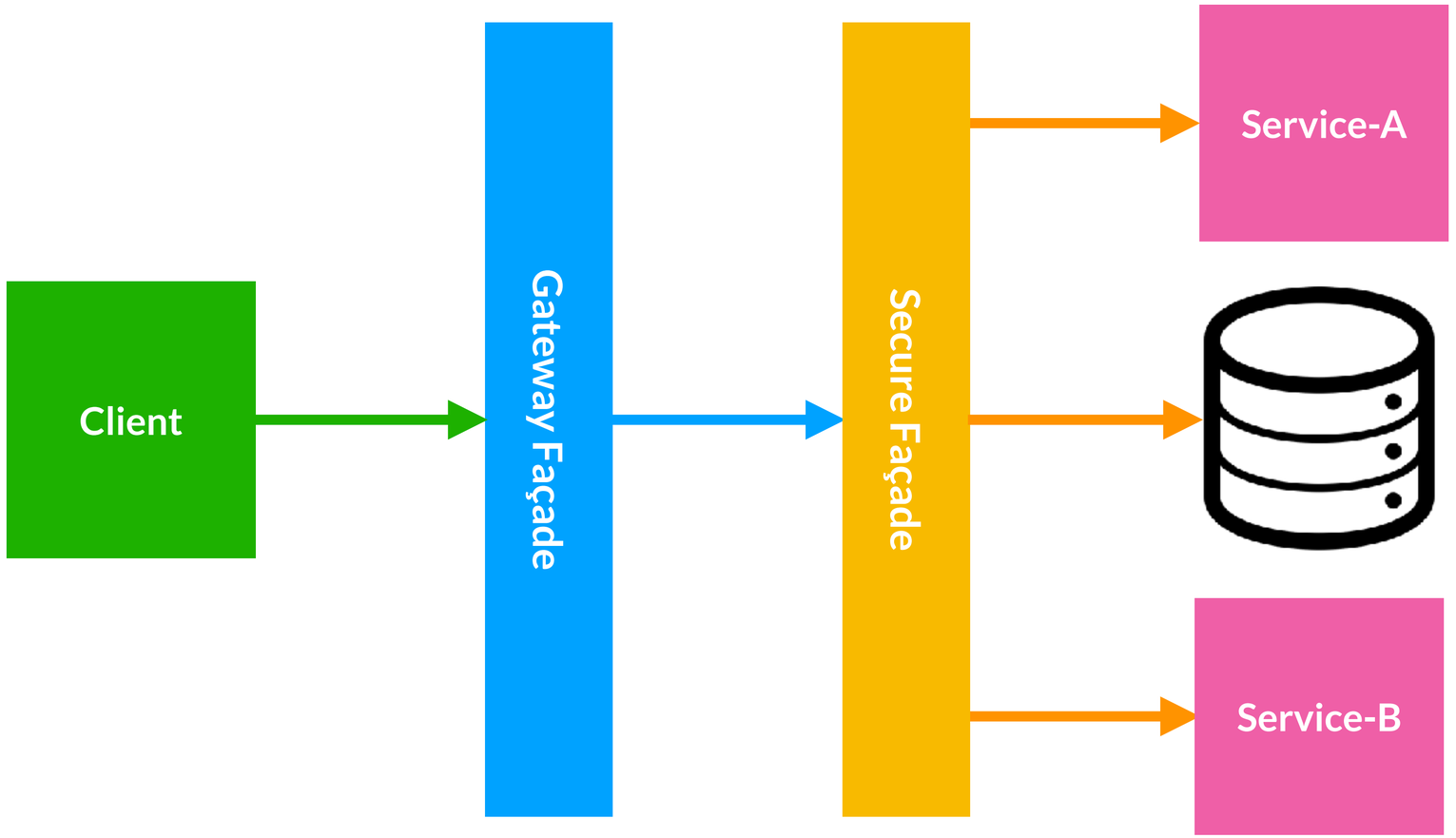
The Solution

- Create a façade that will be entry point of all requests
- The Gateway facade will be in charge of:
  - Accepting Requests
  - Sanitizing the Requests
  - Validating the Requests
  - Mutating the Requests

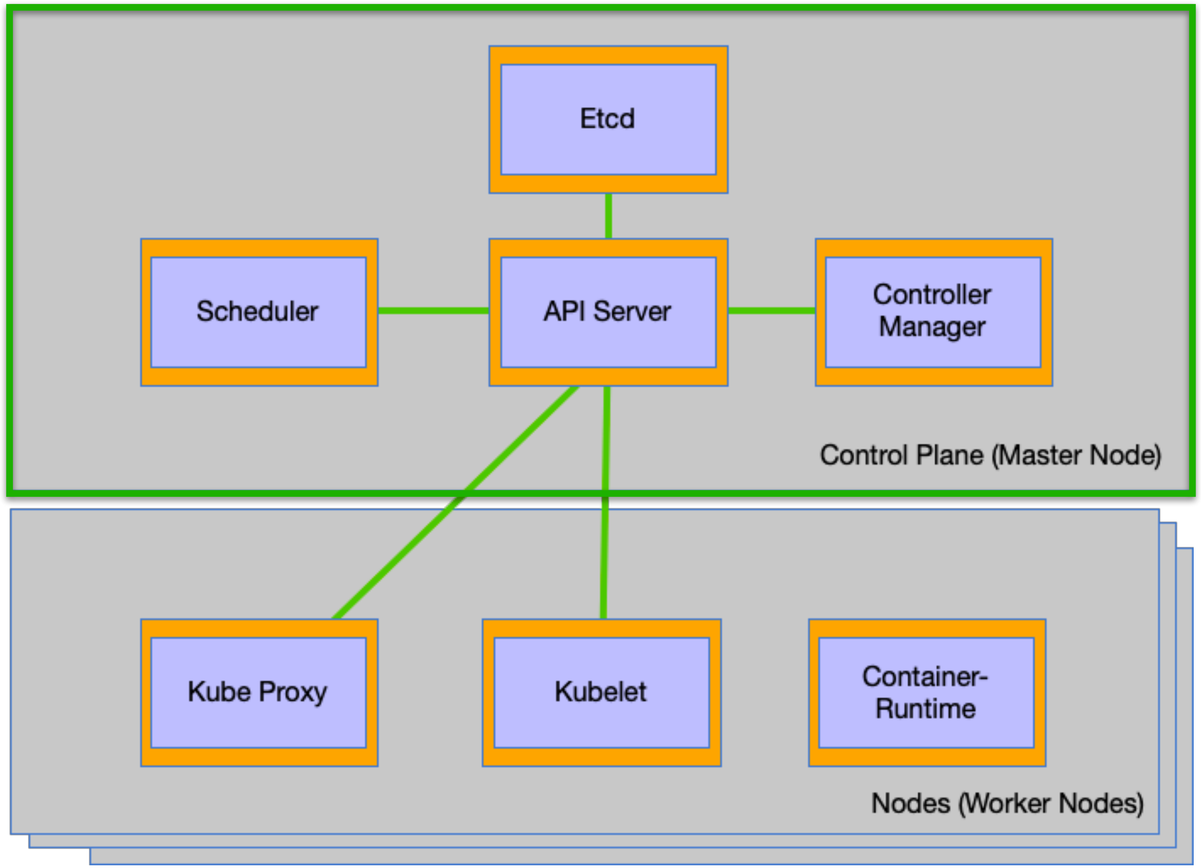
The Diagram

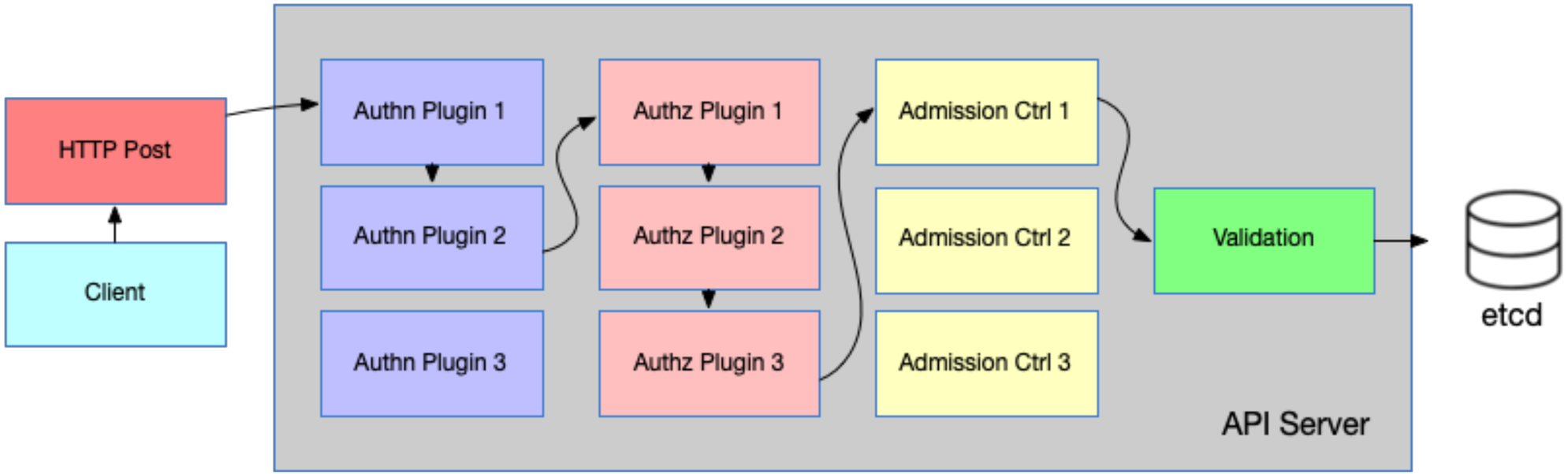


The Diagram

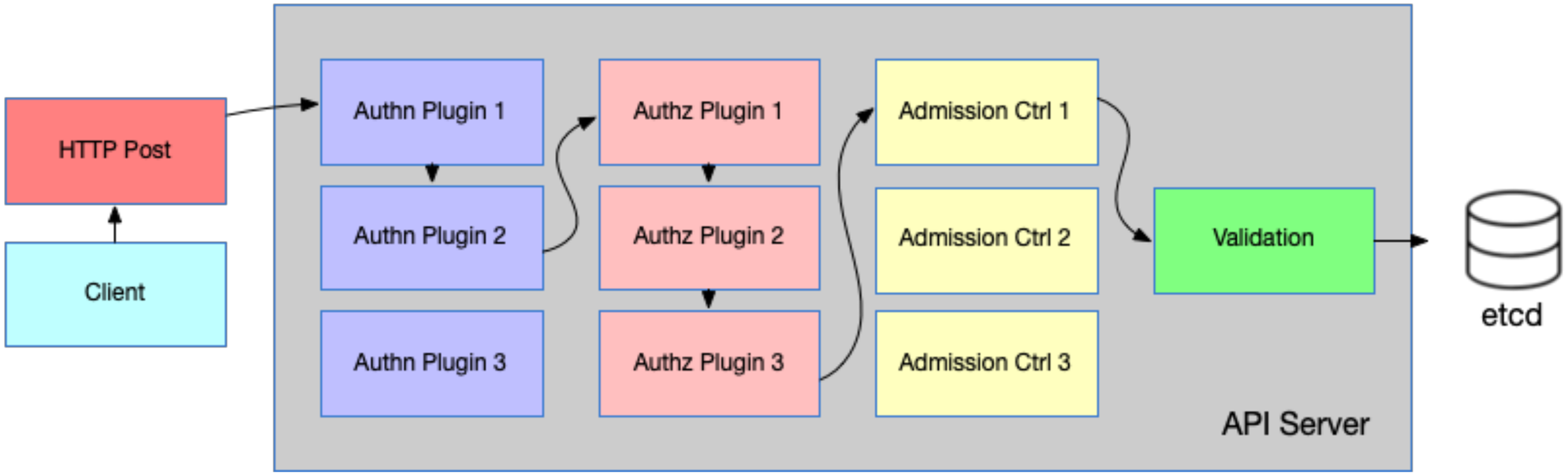


Case Study: Kubernetes





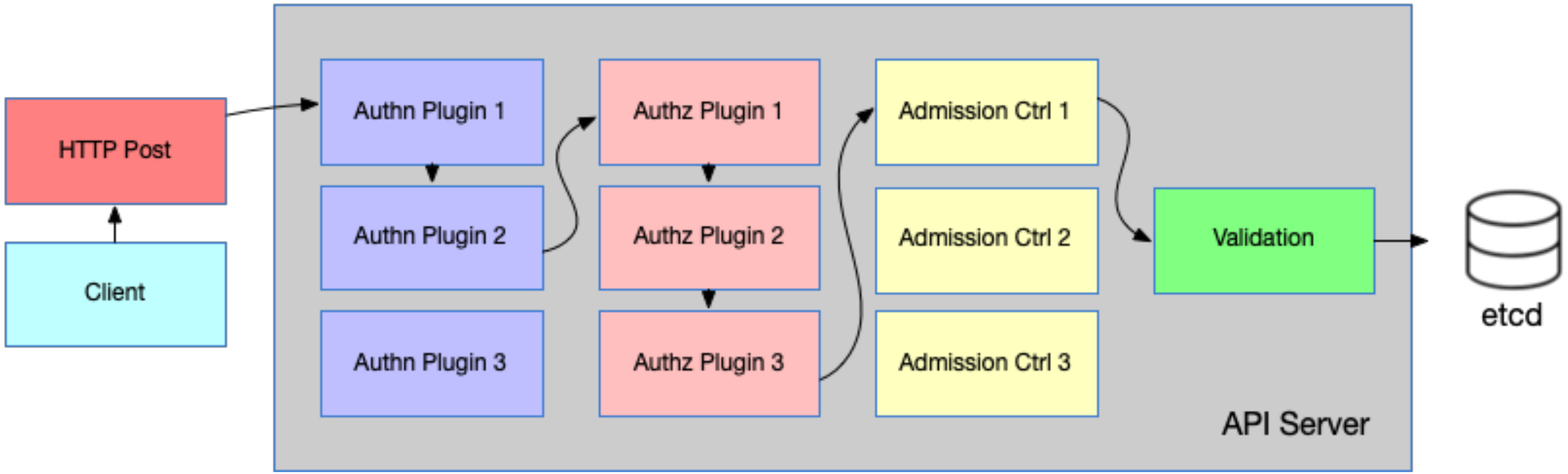
- Client call attempt is made as an HTTP call to the API Server, this includes what we do with `kubectl`
- API Server is RESTful: `get`, `post`, `put`, `patch`, etc.
- The API Authenticates, Authorized, Admit, and Validates before changing etcd



- Authentication is a list of the Authenticated Plugins that identifies who is gaining access to the Kubernetes API
- Authentication Plugins works like a linked list where one will be able to authenticate
- Once established then Authorization Plugins are negotiated

Authentication Plugins

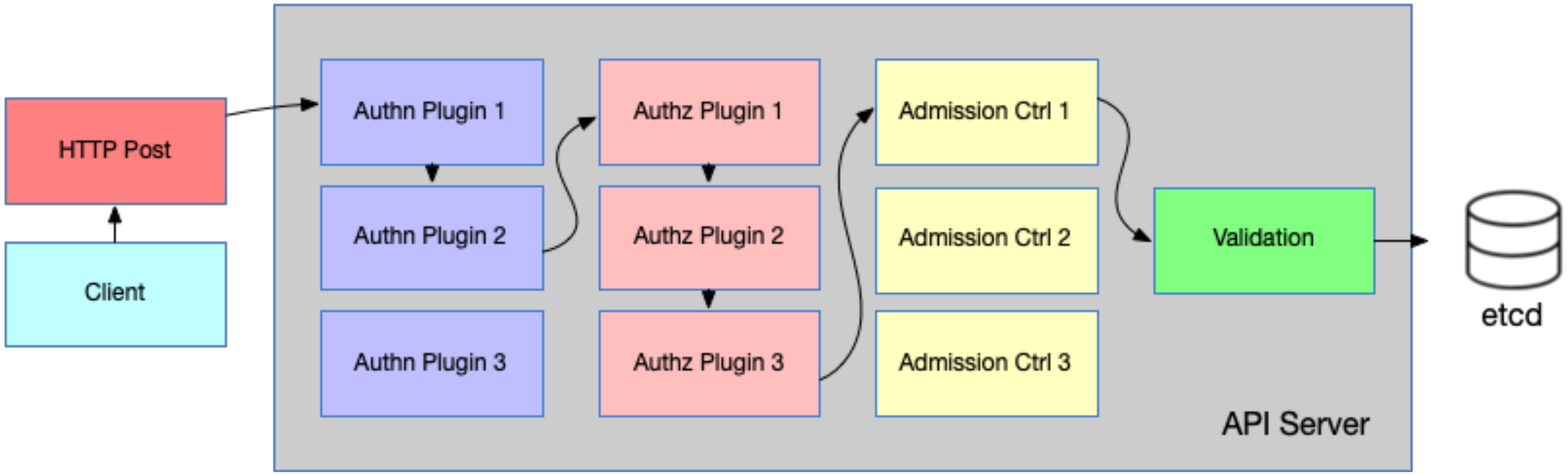
- X509 Client Certs
- Static Token File
- Bearer Tokens
- Bootstrap Tokens
- Service Account Tokens
- Open ID Connect Tokens
- Webhook Token Authentication



- Authorization Plugins works like a linked list where they have plugins
- Given the action that the user wishes to perform, the plugins will determine if the user is allowed to do so.
- As soon as a plugin says the user can perform the action, the API server progresses to the next stage

Authorization Plugins

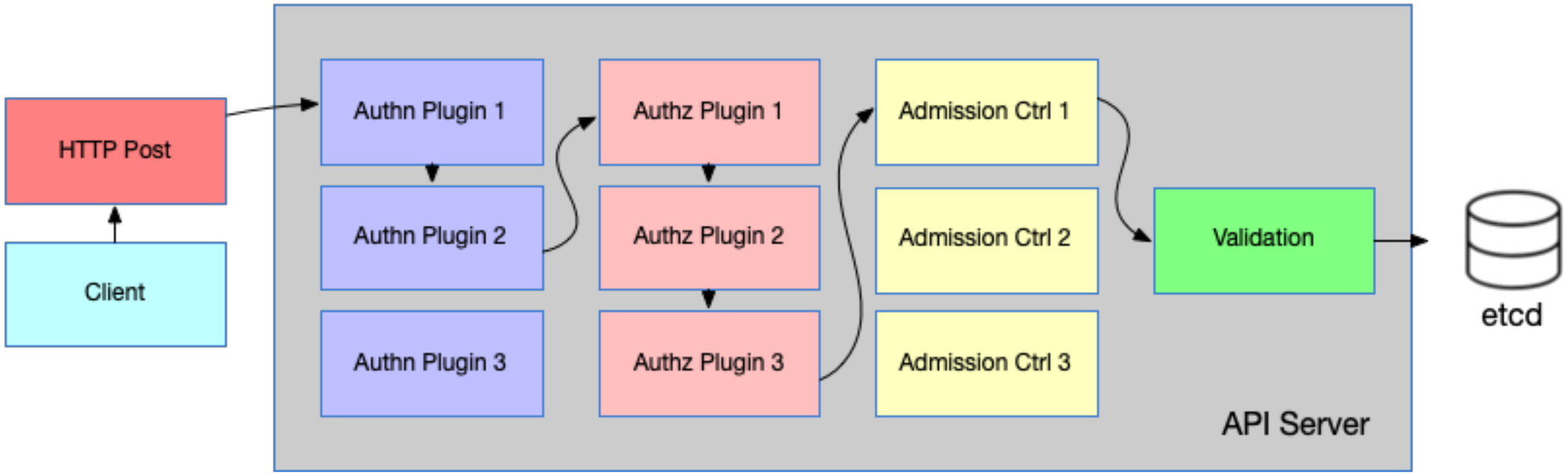
- RBAC Plugin - Role Based Access Control, checks whether an action is allowed to be performed by the user requesting the action
- ABAC Plugin - Attribute Based Access Control, defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together
- Node Plugin - Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets
- Webhook Plugin - WebHook is an HTTP callback mode that allows you to manage authorization using a remote REST endpoint



- Admission Control plugins can modify the resource for different reasons
- They may initialize fields missing from the resource specification to the configured default values or even override them
- They may even modify other related resources, which aren’t in the request, and can also reject a request for whatever reason
- GET (Read) calls don't go through the admission control plugins

Admission Control Plugins

- `AlwaysPullImages`—Overrides the pod's `imagePullPolicy` to `Always`, forcing the image to be pulled every time the pod is deployed.
- `ServiceAccount`—Applies the default service account to pods that don't specify it explicitly.
- `NamespaceLifecycle`—Prevents creation of pods in namespaces that are in the process of being deleted, as well as in non-existing namespaces.
- `ResourceQuota`—Ensures pods in a certain namespace only use as much CPU and memory as has been allotted to the namespace.

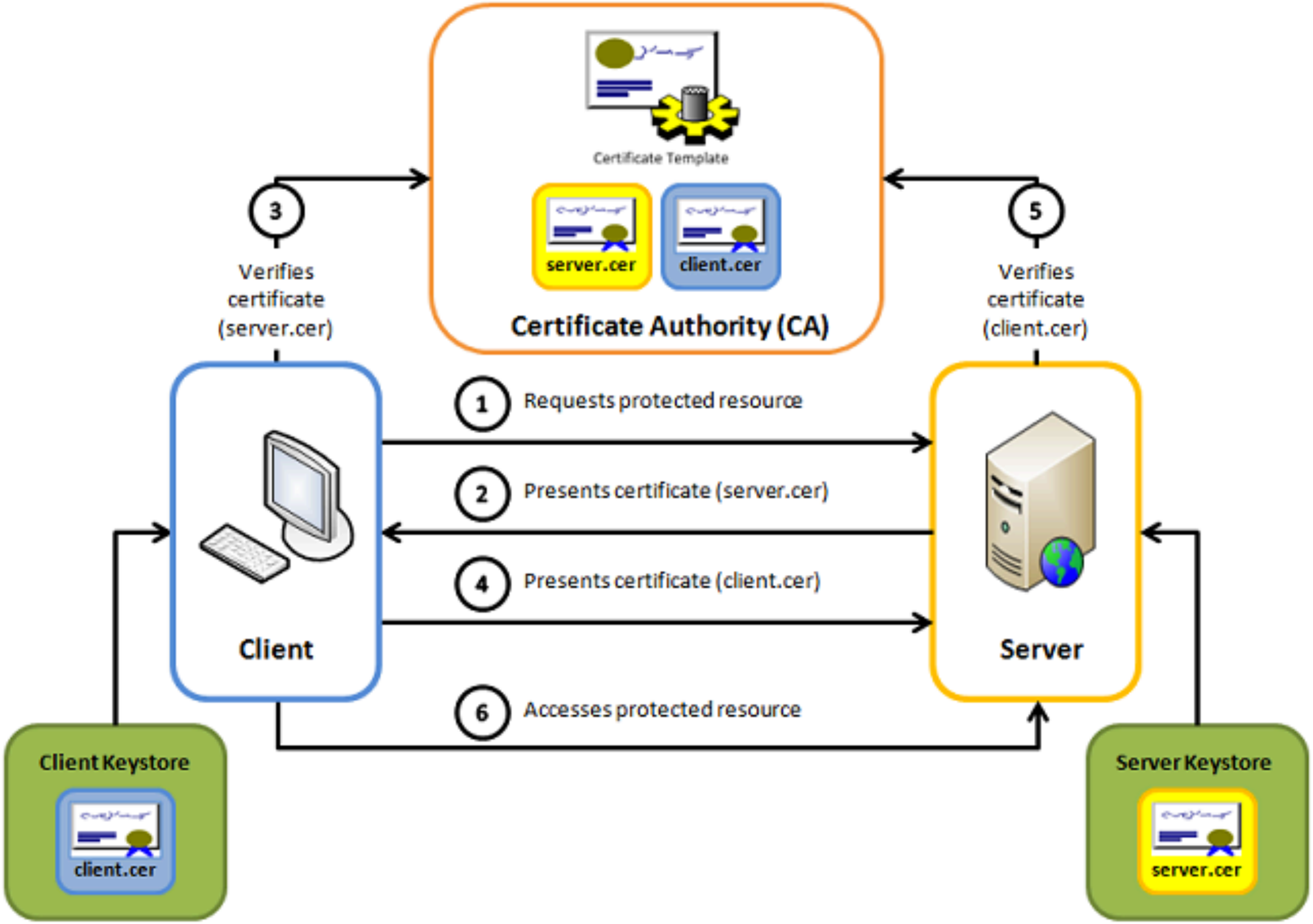


- API server then validates the object, stores it in etcd, and returns a response to the client.
- Validation ensures that the data structure is correct before placing into etcd

mTLS

mTLS Defined

Mutual TLS (mTLS) - Authentication ensures that traffic is both secure and trusted **in both directions** between a client and server.



Mutual SSL authentication / Certificate based mutual authentication

Mutual Authentication flow from codeproject.com

Step for mTLS

- Setup an Ingress Controller
- Create Certificates
- Create Kubernetes Secrets
- Deploy your Application

https://awkwardferny.medium.com/configuring-certificate-based-mutual-authentication-with-kubernetes-ingress-nginx-20e7e38fdfca

Creating the Certificates

- **CommonName(CN):** Identifies the hostname or owner associated with the certificate.
- **Certificate Authority(CA):** A trusted 3rd party that issues Certificates. Usually you would obtain this from a trusted source, but for this example we will just create one. The CN is usually the name of the issuer.
- **Server Certificate:** A Certificate used to identify the server. The CN here is the hostname of the server. The Server Certificate is valid only if it is installed on a server where the hostname matches the CN.
- **Client Certificate:** A Certificate used to identify a client/user. The CN here is usually the name of the client/user.

https://awkwardferny.medium.com/configuring-certificate-based-mutual-authentication-with-kubernetes-ingress-nginx-20e7e38fdfca

Creating the mTLS Certificates

```
# Generate the CA Key and Certificate
$ openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes -subj '/CN=Fern Cert Authority'
# Generate the Server Key, and Certificate and Sign with the CA Certificate
$ openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj '/CN=meow.com'
$ openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt
# Generate the Client Key, and Certificate and Sign with the CA Certificate
$ openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj '/CN=Fern'
$ openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 02 -out client.crt
```

Storing CA and Server CRT

```
$ kubectl create secret generic my-certs --from-file=tls.crt=server.crt --from-
file=tls.key=server.key --from-file=ca.crt=ca.crt

$ kubectl get secret my-certs
NAME       TYPE     DATA   AGE
my-certs   Opaque   3      1m
```

- Deploy your application as normal with Services and Deployments
- We will configure the Ingress to use my-certs as to where the certificate authority and server certificates will reside

Establishing Ingress Certificates

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/auth-tls-verify-client: \"on\"
    nginx.ingress.kubernetes.io/auth-tls-secret: \"default/my-certs\"
  name: meow-ingress
  namespace: default
```

- TLS is enabled and it is using the `tls.key` and `tls.crt` provided in the my-certs secret.
- The `nginx.ingress.kubernetes.io/auth-tls-secret` annotation uses `ca.crt` from the `my-certs` secret.

https://awkwardferny.medium.com/configuring-certificate-based-mutual-authentication-with-kubernetes-ingress-nginx-20e7e38fdfca

Testing mTLS

```
$ curl https://meow.com/ -k
...
<center><h1>400 Bad Request</h1></center>
<center>No required SSL certificate was sent</center>
....
```

- -k is insecure, don't consult with a certificate verification
- In the following, a client certifcation and client key is used to get the payload

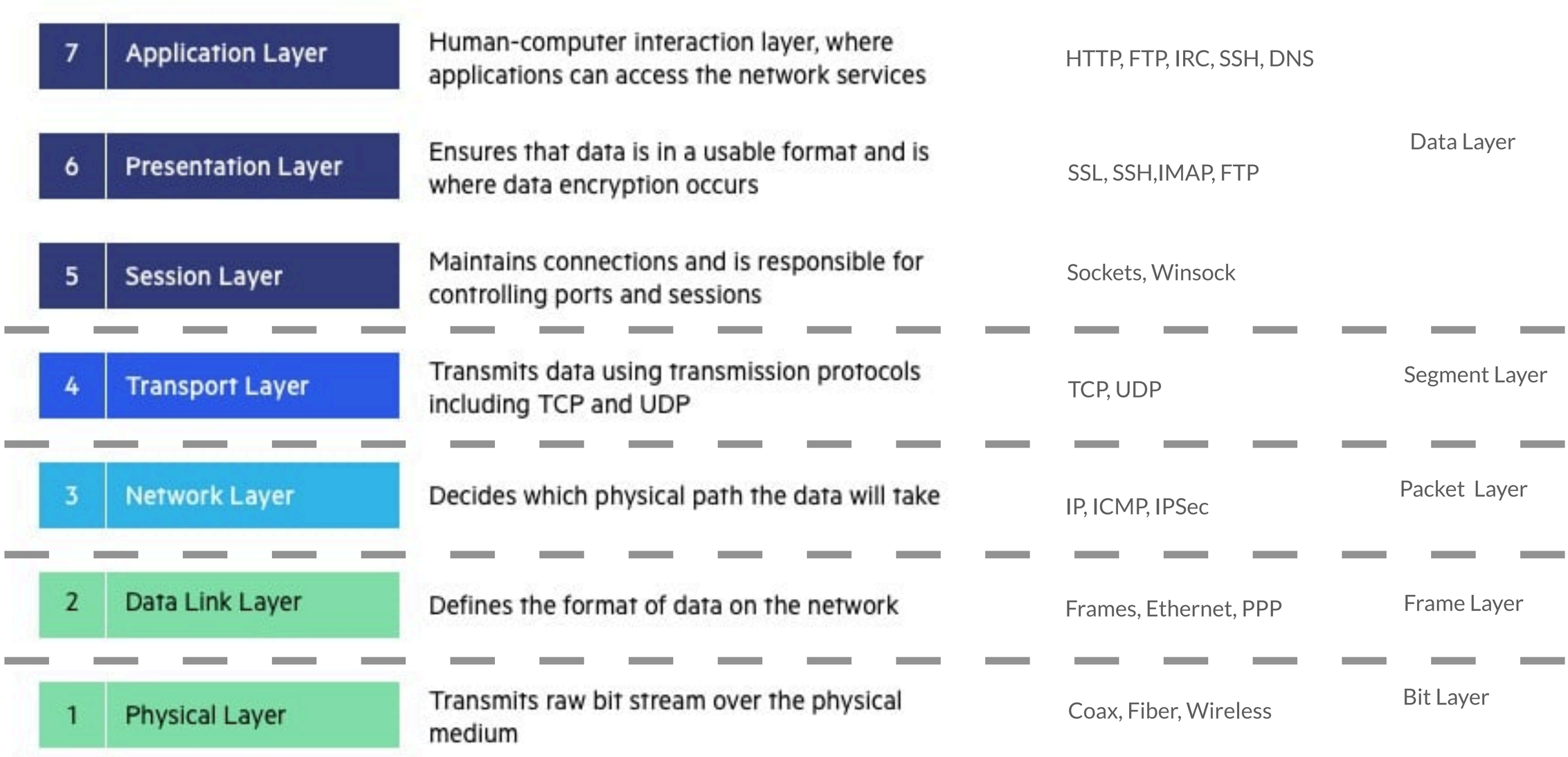
```
$ curl https://meow.com/ --cert client.crt --key client.key -k
...
ssl-client-issuer-dn=CN=Fern Cert Authority
ssl-client-subject-dn=CN=Fern
ssl-client-verify=SUCCESS
user-agent=curl/7.54.0
...
```

https://awkwardferny.medium.com/configuring-certificate-based-mutual-authentication-with-kubernetes-ingress-nginx-20e7e38fdfca

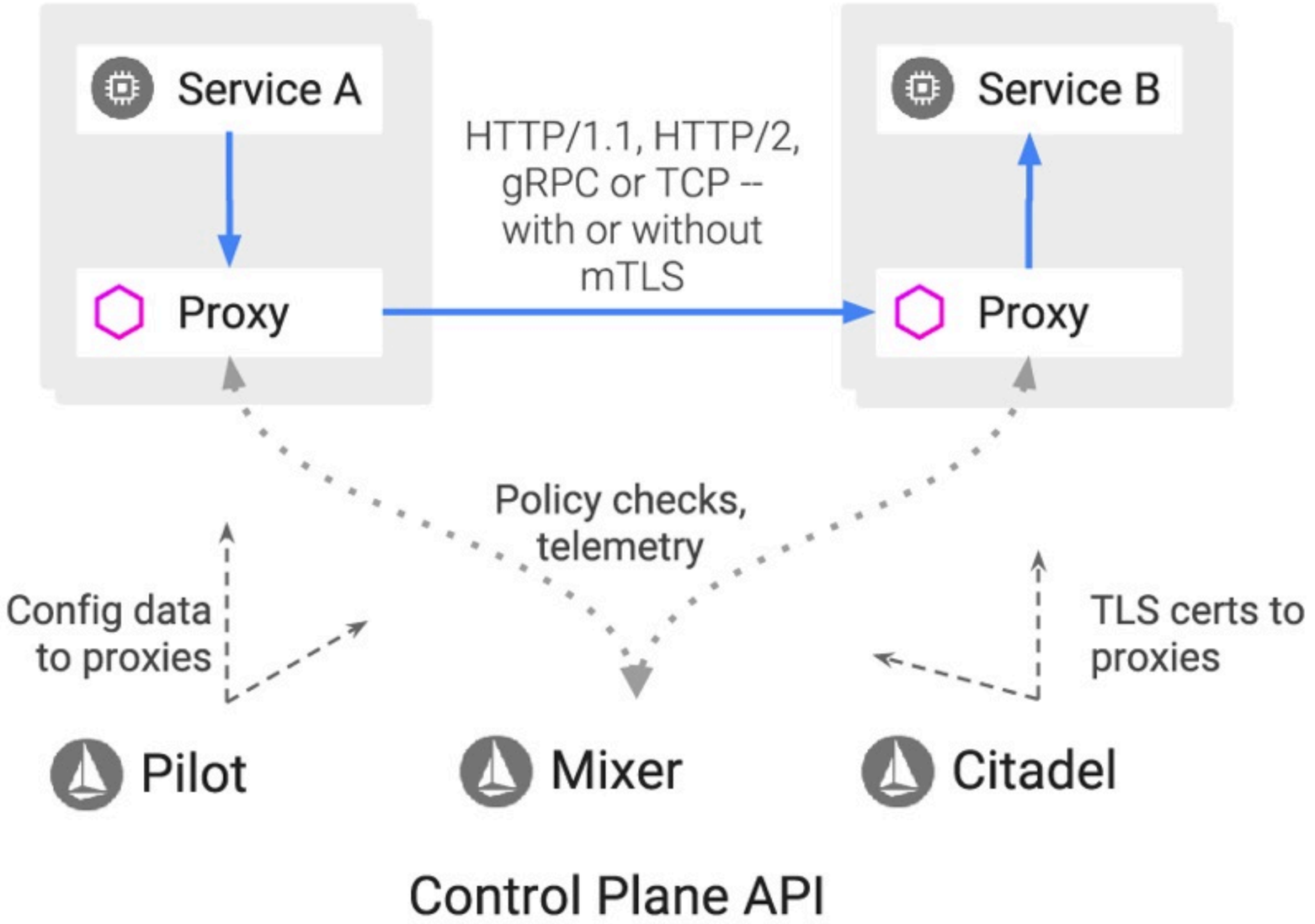
Service Meshes

- A service mesh manages all service-to-service communication within a distributed (potentially microservice-based) software system
- Use of “sidecar” proxies that are deployed alongside each service through which all traffic is transparently routed.
- OSI Layer 7 = Communication using HTTP, now any underlying layer like packets TCP, etc
- Dynamic service discovery and traffic management
- Traffic Shadowing for Testing, Traffic Spitting for Canary
- Including but not limited to **Security Enforcement**!
- Linkerd, Istio, Consul, Kuma, Maesh, AWS App Mesh

OSI Layer Model




Istio


Service A
Service B
HTTP/1.1, HTTP/2,
gRPC or TCP --
with or without
mTLS
Proxy
Proxy
Policy checks,
telemetry
Config data
to proxies
TLS certs to
proxies
Pilot
Mixer
Citadel
Control Plane API

Istio Components

- **Pilot** - Responsible for configuring the Envoy and Mixer at runtime.
- **Proxy / Envoy** - Sidecar proxies per microservice to handle ingress/egress traffic between services in the cluster and from a service to external services.
- **Mixer** - Create a portability layer on top of infrastructure backends. Enforce policies such as ACLs, rate limits, quotas, authentication, request tracing and telemetry collection at an infrastructure level.
- **Citadel / Istio CA -** Secures service to service communication over TLS. Providing a key management system to automate key and certificate generation, distribution, rotation, and revocation.
- **Ingress/Egress** - Configure path based routing for inbound and outbound external traffic.
- **Control Plane API** - Underlying Orchestrator such as Kubernetes or Hashicorp Nomad.

Mutual TLS Authentication

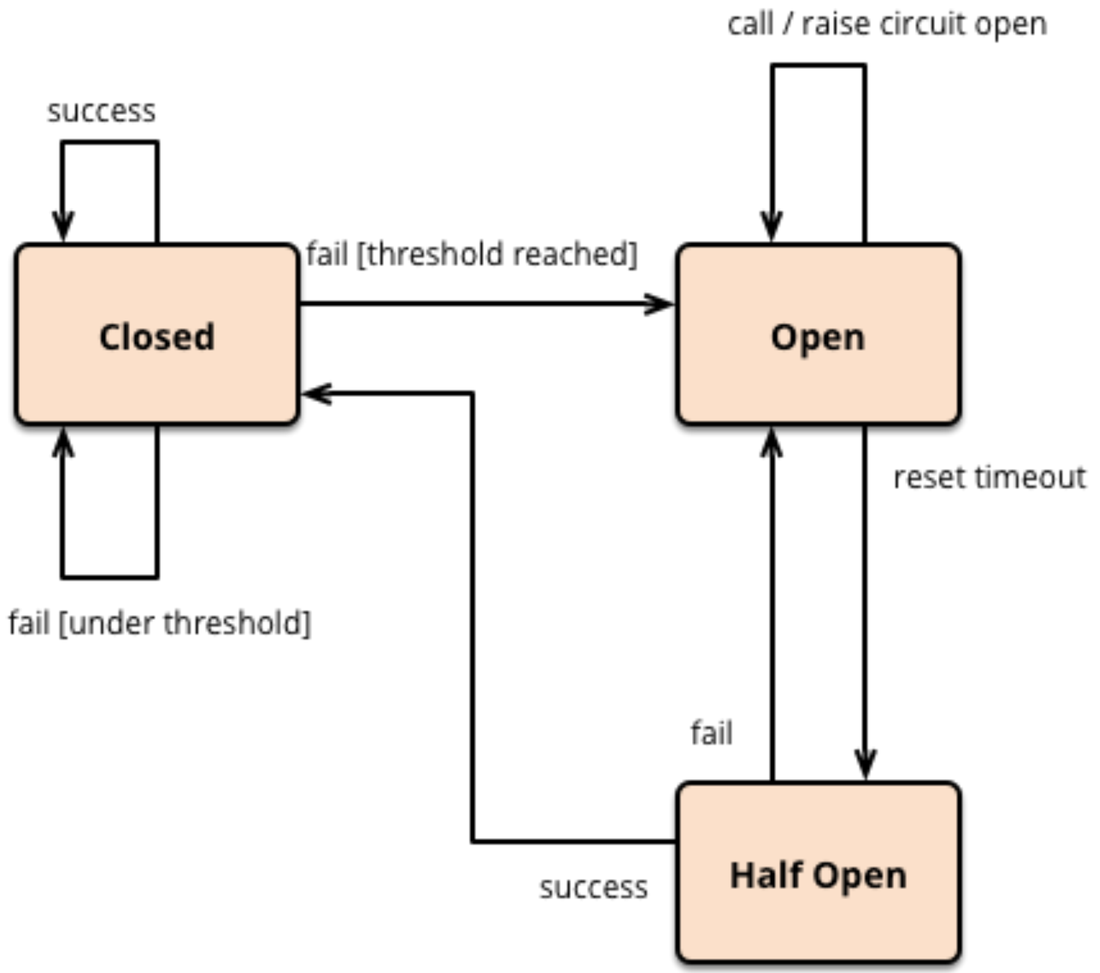
- TLS Communication and Setup performed through Envoy Proxies
- How Istio handles that traffic:
  - Istio re-routes the outbound traffic from a client to the client's local sidecar Envoy.
  - The client side Envoy starts a mutual TLS handshake with the server side Envoy. During the handshake, the client side Envoy also does a secure naming check to verify that the service account presented in the server certificate is authorized to run the target service.
  - The client side Envoy and the server side Envoy establish a mutual TLS connection, and Istio forwards the traffic from the client side Envoy to the server side Envoy.
  - After authorization, the server side Envoy forwards the traffic to the server service through local TCP connections.

https://istio.io/latest/docs/concepts/security/

Ambassador Pattern

The Problem

- Applications require a multitude of services like
  - Circuit breaking
  - Monitoring
  - Metering
  - Datasource Connectivity and Proxying
- There are risks when it comes to application you are unfamiliar with.


call / raise circuit open
success
fail [threshold reached]
Closed
Open
reset timeout
fail [under threshold]
fail
success
Half Open

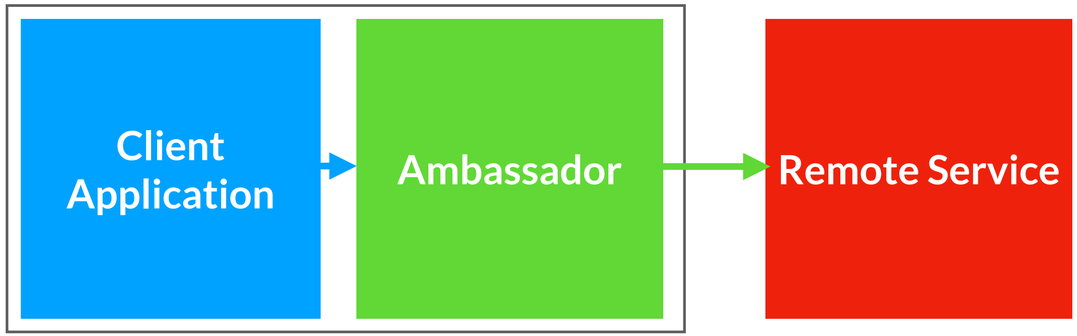
The Solution

- An Ambassador pattern is a proxy for your application to external services
- Deploy the proxy to the same environment, e.g. same pod
- The proxy can monitor performance metrics such as latency or resource usage, and this monitoring happens in the same host environment as the application
- You can make updates to the ambassador without affecting the legacy application
- This can be typically done as a sidecar in the same pod

The Solution

- There is minimal latency overhead
- The solution is better off to be integrated to the application itself
- When client connectivity features are consumed by a single language, a better option might be a client library that is distributed to the development teams as a package

The Diagram



The ambassador will be intercept traffic and provide services for caching, security, circuit breaking, and more

Ambassador vs Sidecar

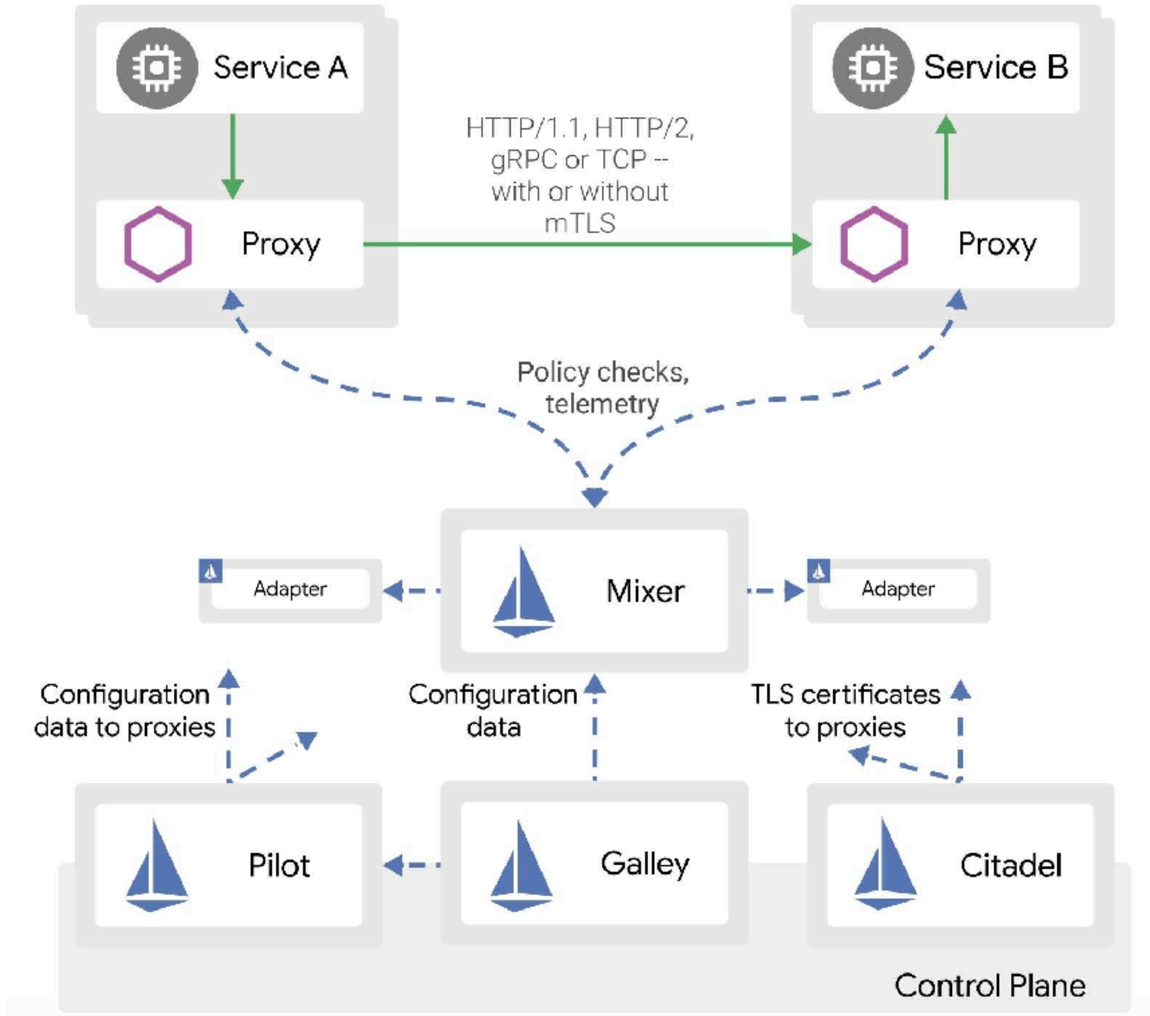
Ambassador:

- Used for monitoring, logging, routing, security (like TLS), all *networking based tasks*
- Used to provide circuit breaking
- Ambassador services can be deployed as a sidecar to accompany the lifecycle of a consuming application or service

Side Car:

- Additional container attached to a parent application and provides supporting features for the application
- If it includes networking it would be considered an ambassador as well
- If it purely logging, configuration it is typically just a sidecar
- It can get rather blurry when trying to distinct between the two

Istio


Service A
Service B
HTTP/1.1, HTTP/2,
gRPC or TCP --
with or without
mTLS
Proxy
Proxy
Policy checks,
telemetry
Adapter
Mixer
Adapter
Configuration
data to proxies
Configuration
data
TLS certificates
to proxies
Pilot
Galley
Citadel
Control Plane

Istio Components

- **Pilot** - Responsible for configuring the Envoy and Mixer at runtime.
- **Proxy / Envoy** - Sidecar proxies per microservice to handle ingress/egress traffic between services in the cluster and from a service to external services.
- **Mixer** - Create a portability layer on top of infrastructure backends. Enforce policies such as ACLs, rate limits, quotas, authentication, request tracing and telemetry collection at an infrastructure level.
- **Citadel / Istio CA -** Secures service to service communication over TLS. Providing a key management system to automate key and certificate generation, distribution, rotation, and revocation.
- **Ingress/Egress** - Configure path based routing for inbound and outbound external traffic.
- **Control Plane API** - Underlying Orchestrator such as Kubernetes or Hashicorp Nomad.

Demo: Envoy

- Envoy was the basis for Istio
- With envoy you can set up proxies to perform a variety of actions
- View the documentation here: https://www.envoyproxy.io/docs/envoy/v1.29.2/

Shift Left

Shift Left

- “Go Early” - If the action you need done, particularly security, do that first
- Begins on the left and finish on the right.
- Tasks that need to be planned for, like test driven development, move that set of tasks to the left…earlier in the process.
- Things like testing, privacy by design and default, security, all shift left in the process, instead of trying to bolt it on at the end of the process.
- The boundary is going to be before your first staging deployment

What are some Shift Left activities?

- Continuous Integration
- Security Scanning
- Static Code Analysis
- Threat Modeling
- Architecture Planning
- Architecture Overview
- Code Signing
- Byte Code Scanning

Small Images

Center for Internet Security

Home > **CIS Benchmarks**

CIS Benchmarks List

The CIS Benchmarks™ are prescriptive configuration recommendations for more than 25+ vendor product families. They represent the consensus-based effort of cybersecurity experts globally to help you protect your systems against threats more confidently.

DOWNLOAD BENCHMARKS →

Small Images

From the CIS Benchmark:

- Use multi-stage builds to produce a final image as clean as possible
- Pay close attention to the RUN commands when reviewing Dockerfiles
- Make sure Dockerfiles don’t mount sensitive directories like /etc or /bin
- Don’t include sensitive data in the Dockerfile
- Include everything that your container needs (immutable containers)
- Try to use digest tags instead of canonical versioning (protect from tampering and corruption)

Small Images

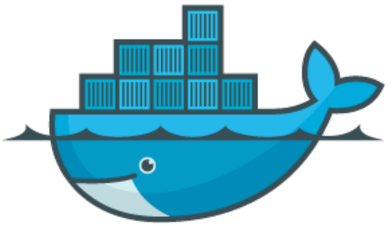
```
FROM golang:1.14.4 as builder
WORKDIR /app/
COPY *.go ./
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .

FROM alpine:3.12.0
WORKDIR /root/
RUN apk --no-cache add ca-certificates
COPY --from=builder /app/main .
CMD ["./main"]
```

Go Example of Multistage Build

How is it done in Java?

- Let’s take a look at https://levelup.gitconnected.com/java-developing-smaller-docker-images-with-jdeps-and-jlink-d4278718c550



Demo: Small Images

- Let’s create a small Java based image using Java Modules (JDK 9+)
- Slimming your docker image lessens your attack surface

Certificate Management

Keys

- Symmetric Keys:
  - Use the same key for both encryption and decryption.
  - Fast and efficient for large data.
  - Examples include AES (Advanced Encryption Standard).
- Asymmetric Keys:
  - Use a pair of keys: a public key and a private key.
  - Public key encrypts data, private key decrypts it, and vice versa.
  - More secure but slower compared to symmetric keys.
  - Examples include RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography).

Certificates

- A certificate contains a public key.
- The certificate, in addition to containing the public key, contains additional information such as:
  - The issuer
  - The purpose of the certificate is supposed to be used
  - The Certificate Authority
  - Validity Period
  - Other Metadata

Certificate Authority

- A certificate authority is a company or organization that acts to validate the identities of entities (such as websites, email addresses, companies, or individual persons) and bind them to cryptographic keys through the issuance of electronic documents known as *digital certificates*.

- Can be a public CA (trusted by browsers and operating systems) or a private CA (used within an organization

- For Example, every Kubernetes Cluster has its own CA. The CA is generally used by cluster components to validate the API server's certificate, by the API server to validate kubelet client certificates, etc.

Certificate Rotation

- Certificates should be rotated periodically, to ensure optimal security.
- Certificate rotation (which means the replacement of existing certificates with new ones) is needed when:
  - Any certificate expires.
  - A new CA authority is substituted for the old; thus requiring a replacement root certificate for the cluster.
  - New or modified constraints need to be imposed on one or more certificates.
  - A security breach has occurred, such that existing certificate-chains can no longer be trusted.

Example Process

1. Server generates an asymmetric key pair (public and private keys).
2. Server creates a CSR with its public key and identity details (e.g., domain name).
3. Server submits the CSR to a CA.
4. CA verifies the server’s identity (e.g., domain ownership).
5. CA issues a certificate by signing the CSR with the CA’s private key.
6. Server installs the certificate and corresponding private key.
7. Client connects to the server, receiving the server’s certificate.
8. Client verifies the certificate using the CA’s public key.

Secret Management

Secret Management

- Know how secrets are being stored
- For example: By default, Kubernetes etcd is unencrypted by default
- You can use Hashicorp Vault or any of the cloud providers security management like KMS to handle secrets

Valet Key

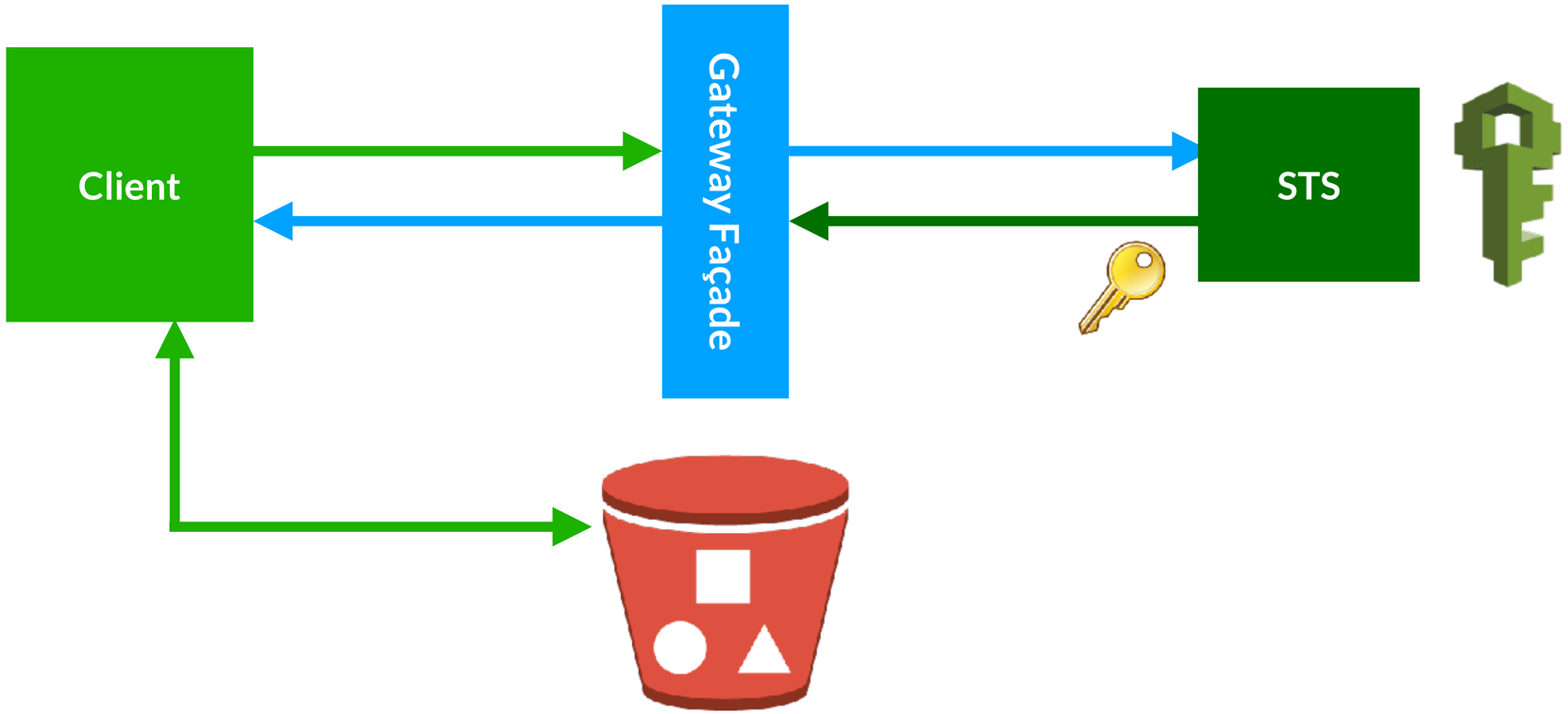
The Problem

- Services may:
  - Reach into a storage account and stream it to a client
  - Accept a payload from a client and store it into a storage account
- Each of the scenarios may present an unnecessary middleman
- We can provide any client access to the datastore without interaction to the service, but that means that the service will not longer act as *Gatekeeper*
- *A*pplications must be able to securely control access to data in a granular way

The Solution

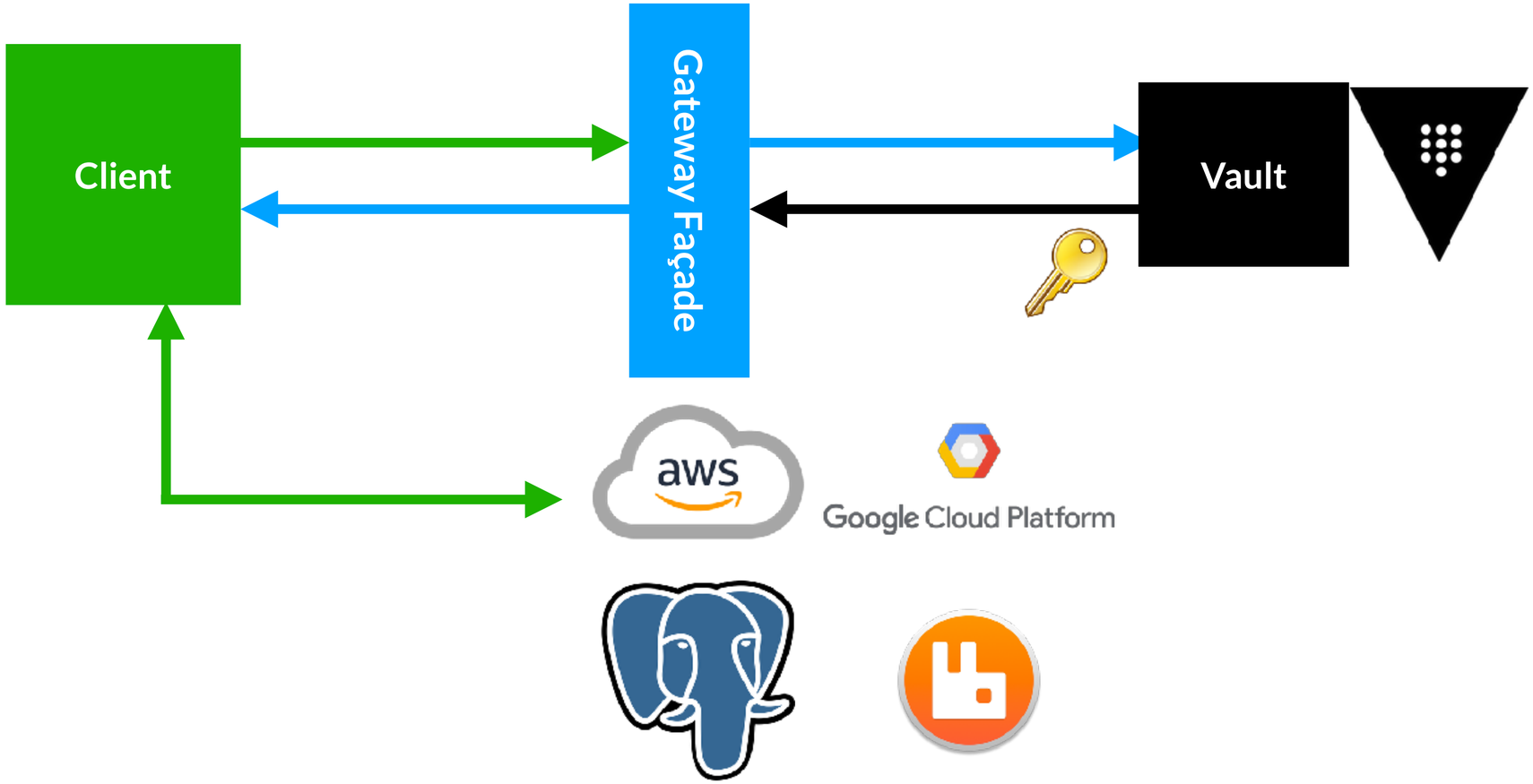
- Restrict access to the data store's public connection
- Provide the client with a key or token that the data store can validate.
- The token is referred to as a valet key
- Valet key has a time to live
- Contains access permissions like roles, which can be specific

The Diagram



Amazon Security Token Service
https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html

The Diagram with Vault



https://www.vaultproject.io/docs/secrets/aws

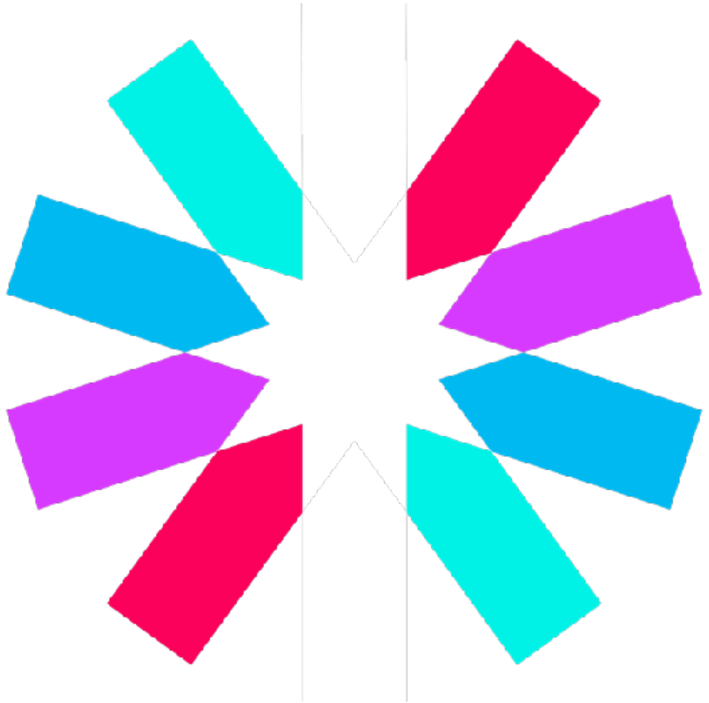
Demo: Valet Key

- We will use Vault to generate tokens to use with a PostgreSQL database
- We will initialize the connection to Vault
- Set up a connection and role
- Get a temporary credential to the database which a client can use to gain direct access.

JSON Web Token

JSON Web Token

- Internet standard for creating data with
  optional signature and/or
  optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.
- Claims are common way for applications to acquire the identity information they need about users inside their organization, in other organizations, and on the Internet.
- A server could generate a token that has the claim "logged in as administrator" and provide that to a client. The client could then use that token to prove that it is logged in as admin.



https://en.wikipedia.org/wiki/JSON_Web_Token

What was wrong with Session Cookies

- Cookies are/were common: Client would authenticate to the server, server would respond with a cookie
- Common to be attacked: Cross-Site-Request-Forgery
- Session would need to be stored in a database or memory (like your web server). This becomes an issue with microservices and load-balancer

https://en.wikipedia.org/wiki/JSON_Web_Token

Typical JWT Claims

The JWT specification defines seven reserved claims that are not required, but are recommended to allow interoperability with third-party applications. These are:

- iss (issuer): Issuer of the JWT
- sub (subject): Subject of the JWT (the user)
- aud (audience): Recipient for which the JWT is intended
- exp (expiration time): Time after which the JWT expires
- nbf (not before time): Time before which the JWT must not be accepted for processing
- iat (issued at time): Time at which the JWT was issued; can be used to determine age of the JWT
- jti (JWT ID): Unique identifier; can be used to prevent the JWT from being replayed (allows a token to be used only once)

Full list of claims are here: https://www.iana.org/assignments/jwt/jwt.xhtml#claims

How is the Token Transported

How are JWT Used:

- JWT tokens shall be transmitted over TLS, such that nobody else can read it (and impersonate the user). So for a secure JWT token usage, ***HTTPS is a prerequisite***.
- The server can use JWT's payload to determine who the caller is and what they can do. It's the only way to tell individual users/requests apart from the server's perspective.
- JWT tokens also support asymmetric signatures, such that server A can sign a JWT token, server B can choose to trust server A's public keys and therefore validate JWT tokens issued by server A.
- `Authentication: Bearer <jwt-token>` The server takes this header, unwraps the JWT token, determines who the caller is and whether the request shall pass through, and only then acts on the actual request content.

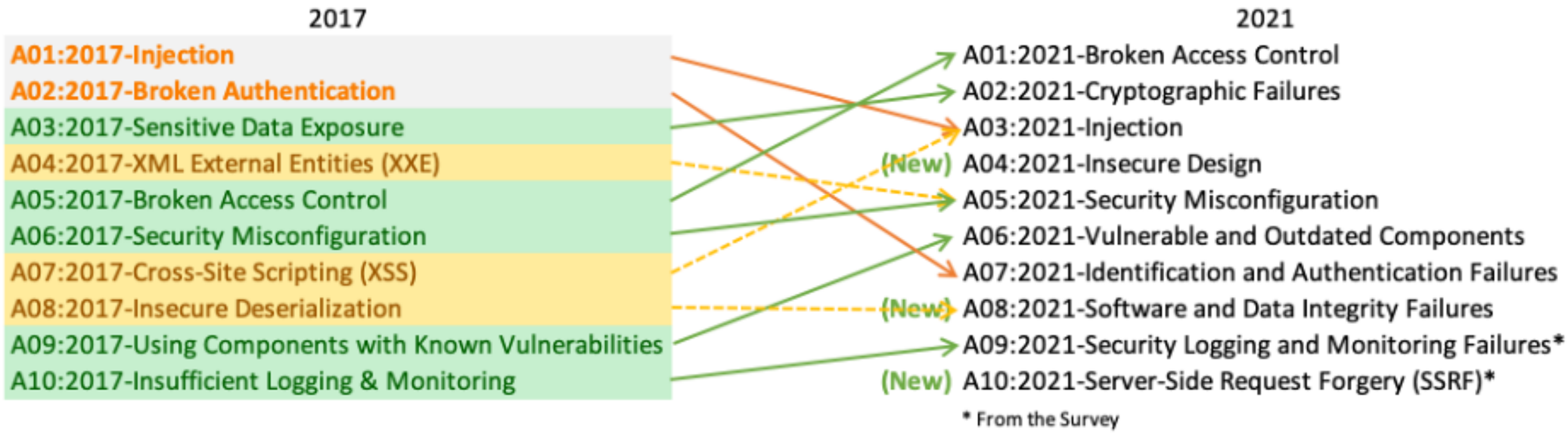
Demo: JWT

- Let’s take a look at a JSON Web Token
- We will use their website at jwt.io to create a key, and use a secret to help validate the JWT and identity

OWASP

OWASP

- The Open Worldwide Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software.
- Includes Security Listings, Research and Products



https://owasp.org/www-project-top-ten/

Securing Supply Chains

Software Bill of Materials

- Key building block in software security and software supply chain risk management
- An SBOM is a nested inventory, a comprehensive list of all the software components, dependencies, and metadata associated with an application

Why is it required?

- The need for SBOMs is driven by several factors that include:
  - Ensuring software transparency
  - Managing open-source software and third-party dependencies
  - Identifying and mitigating security vulnerabilities
  - Complying with legal and regulatory requirements
- Part of U.S. Executive Order on improving the nation’s cybersecurity

What do the SBOMs look like?

- SBOMs have three different “standards”
  - CycloneDX
  - SPDX
  - SWID Tags
- Which one is popular?
  - Cyclone DX by OWASP
  - SPDX is by Linux Foundation and is more of the open standard
  - SWID Tags from the NIST (National Institute of Standards and Technology) aren’t necessarily SBOMs but tag data

What Software is Available?

- CycloneDX
- SPDX
- SWID
- OWASP Dependency-Track
- FOSSA
- Black Duck
- Snyk
- White-Source

Demo: SBOMs and Dependency Check

- Let’s integrate an SBOM in a Java Project
- Followed up with Running Dependency Check in Jenkins

RBAC

RBAC

- Role Based Access Control
- Most System will typically have a Role Based Access Control, based on an identity and what group they belong
- Susan is with the group engineering, then let’s secure this resource where we only allow engineers into this resource
- Susan is not directly provided access to the resource, but indirectly through her role.

Essential Tools

KubeBench

- https://github.com/aquasecurity/kube-bench
- kube-bench is a tool that checks whether Kubernetes is deployed securely by running the checks documented in the CIS Kubernetes Benchmark

```
$ kubectl apply -f job.yaml
job.batch/kube-bench created

$ kubectl get pods
NAME                       READY   STATUS              RESTARTS   AGE
kube-bench-j76s9   0/1     ContainerCreating   0          3s

# Wait for a few seconds for the job to complete
$ kubectl get pods
NAME                       READY   STATUS      RESTARTS   AGE
kube-bench-j76s9   0/1     Completed   0          11s

# The results are held in the pod's logs
kubectl logs kube-bench-j76s9
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
...
```

Tracee

- Observe system calls made by your container
- Uses eBPF as a sandbox and determine what capabilities your container uses
- Once the report is done, you can remove capabilities from image

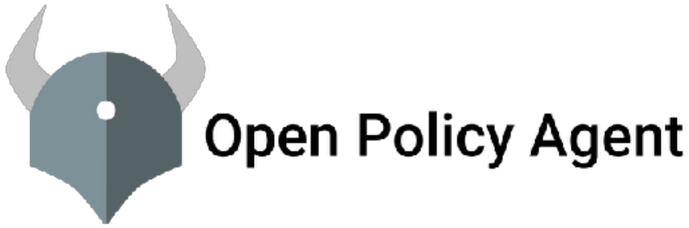
```
docker run \
  --name tracee --rm -it \
  --pid=host --cgroupns=host --privileged \
  -v /etc/os-release:/etc/os-release-host:ro \
  -e LIBBPFGO_OSRELEASE_FILE=/etc/os-release-host \
  aquasec/tracee:latest
```

```
docker run -it --rm -p 8080:80 nginx:alpine
```

```
docker run --cap-drop=all --cap-add=<cap1> --cap-add=<cap2> <image> ...
```

Open Policy Agent

- Admission Controllers enforce semantic validation of objects during create, update, and delete operations.
- With OPA you can enforce custom policies on Kubernetes objects without recompiling or reconfiguring the Kubernetes API server
- Uses a configuration language called Rego to enforce those policies
- https://www.openpolicyagent.org/docs/v0.12.2/kubernetes-admission-control/



Trivy

- Trivy has different *scanners* that look for different security issues, and different *targets* where it can find those issues.
- Trivy's targets can include docker containers, file systems, and Kubernetes clusters
- https://aquasecurity.github.io/trivy/v0.28.1/

```
trivy image python:3.4-alpine
```

```
trivy fs --security-checks vuln,secret,config myproject/
```

```
trivy k8s mycluster
```

Kube-Hunter

- kube-hunter hunts for security weaknesses in Kubernetes clusters [Pen Tool]
- The tool was developed to increase awareness and visibility for security issues in Kubernetes environments

```
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-hunter
spec:
  template:
    spec:
      containers:
      - name: kube-hunter
        image: aquasec/kube-hunter
        command: ["kube-hunter"]
        args: ["--pod"]
      restartPolicy: Never
  backoffLimit: 4
```

Falco

- A CNCF incubating project
- Falco is an open source runtime security tool
- Falco parses Linux system calls from the kernel at runtime
- Asserts the stream against a rules engine
- Run as a Kubernetes Job
- If a rule is violated, Falco triggers an alarm

```
helm install falco falcosecurity/falco
kubectl get pods
```

Falco, what does it check by default?

- Privilege escalation using privileged containers
- Namespaces changes using tools like setns
- Read/Write to well-known directories such as /etc, /usr/bin
- Creating symlinks
- Ownership and mode changes
- Executing SSH or shell binaries

Thank You

- Email: dhinojosa@evolutionnext.com
- Github: https://www.github.com/dhinojosa
- Mastodon: https://mastodon.social/@dhinojosa
- Linked In: http://www.linkedin.com/in/dhevolutionnext