API Management Proposal for AWSI

Prepared for
AUDI AG

Ingolstadt, 11.04.2017

Nagarro GmbH

Stefan Böltl,
Heiko Dietze,
Holger Martin

Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- Architecture
- Gateway installation and provisioning
- Realization Proposals
- Summary
- Roadmap
- References

Focus

Scope of Project

Covering the IAA show case eCOM-GW (Gebrauchtwagen-Portal) by:

- Integration of an API Gateway into AWSI
- Establish an provisioning environment for the API Gateway (GMU)
- Securing services by:
  - SAML (user authentication by the iPortal)
  - CrowdID (user authentication by the Atlassian Crowd session)
  - ApiKey (only client credentials)
- Establishing an role mapping between external IdPs and AWSI roles
- Establishing an user info endpoint for AWSI services
- Establishing an concept for automatic service provisioning

Agenda

API Management Proposal for AWSI

- Focus
- **Requirements**
- Use Cases
- Architecture
- Gateway installation and provisioning
- Realization Proposals
- Summary
- Roadmap
- References

Requirements

General Requirements

- Deployment as Docker-Image or AMI
- configuration via cloud-config/cloud-init preferred
- for Docker Container configuration via environment variables from repository
- Provisioning only via Jenkins jobs (Chef not more in place)

Requirements

IAM Requirements

- AWSI IAM provides possibilities to manage AWSI internal roles
  - There are no myAudi Users in AWSI IAM
  - IAM provides a mapping capability for external dealer and importer groups to internal AWSI groups
- AWSI IAM supports only federated authentication
  - AWSI IAM only trust approved identity provider
  - application user or dealer should use there known portal to login
  - AWSI IAM supports OpenIDConnect, SAML, OAUTH2
  - there is no ldap sync
  - there are no passwords in AWSI user data store
- AWSI IAM supports token authentication inside the platform
  - AWSI IAM sends a token with user ID of ID + IDP + AWSI internal roles
- AWSI IAM is responsible for authentication
  - AWSI backend services are responsible for authorization based on AWSI internal roles
  - AWSI API Gateway supports the possibility to set a access control list of an API, but not on the methods of the API
- AWSI backend service decide whether they are exposed in internet (if they get an API inside the API gateway

Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- **Use Cases**
- Architecture
- Gateway installation and provisioning
- Realization Proposals
- Summary
- Roadmap
- References

Use Cases

iPortal user invokes AWSI service - SAML Authentication

Use Case:

An already on the iPortal authenticated user tries to access AWSI services.

Task:

The API gateway verifies the signature of the received SAML authentication assertion. If there is an trusted relation to the issuer (via X.509 certificate) the gateway retrieves the userID and the roles from the assertion. After a role mapping to AWSI roles an JSON Web Token (JWT) is created and signed for the use with the backend services.

Implementation:

Currently it is foreseen that client authenticated by the iPortal use a SAML bearer token. This SAML token is exchanged by an JWT which grants access as defined in RFC 7522 (SAML Bearer Token) and RFC 7519 (JWT).

Fallback strategy :

Use of SAML-Protocol: IdP initiated authentication or SAML-Protocol: SP initiated authentication

Use Cases

AWSI developers invoke AWSI service - Atlassian Crowd Authentication

Use Case:

An already on the AWSI crowd authenticated user (AWSI service developer) tries to access AWSI services.

Task:

The API gateway verifies the token via crowd authentication REST interface. The gateway retrieves the userId (email) and the roles from the crowd. After a role mapping to AWSI roles an JSON Web Token (JWT) is created and signed for the use with the backend services.

Implementation:

The crowd IdP is integrated via its proprietary REST interface.

Use Cases

External service invokes AWSI service – API key authentication

Use Case:

An external service tries to access AWSI services by an API key. The API key is a client credential which identifies the calling service, but not a user. The calling service has to be registered to retrieve an API key. The registration process is manually done in the first step.

Task:

The API gateway verifies the API key. The gateway retrieves the API key and assigns the registered roles. An JSON Web Token (JWT) is created and signed for the use with the backend services.

Implementation:

The API Gateway substitutes the API key with a JWT.

Use Cases

An internal (not user triggered) service invokes AWSI service – API key authentication

Use Case:

An internal service without user context e.g. a time scheduled service tries to access AWSI services.

Task:

The calling service has to use an API key and to call the service via the API gateway. The API gateway reuses the functionality of “external service invokes AWSI service” use case.

Implementation:

The API Gateway substitutes the API key with a JWT.

Use Cases

AWSI service invokes AWSI service – authentication/authorization per JWT

Use Case:

An AWSI service tries to access another AWSI service.

Task:

The calling service passes its JWT to service to be invoked. The invoked service verifies the signature, expiration time (exp) and audience (aud)*. On success it fulfils its tasks under consideration of the passed roles. Otherwise it rejects the request by responding an appropriate http status code.

Implementation:

The behaviour has to be implemented by the AWSI services.

*Audience restriction should be foreseen on the services, but is not implemented in the first step

Use Cases

AWSI service invokes external service – credential exchange

Use Case:

An AWSI service tries to access an external service.

Task:

The calling service invokes a service on an outbound service endpoint on the API Gateway and passes its JWT. The Gateway exchange the credentials . The credentials of the invoked service are managed exclusively on the Gateway.

Implementation:

The credential exchange is done by the API Gateway. *

*This use case is not planned for the IAA

Use Cases

Role mapping between IdP and AWSI

Use Case:

Authorization roles of a certain IdP has to be converted into appropriate AWSI roles.

Task:

During the authentication process the API gateway converts the IdP specific roles to appropriate role within AWSI. The roles information are retrieved from the authenticating token e.g. SAML attribute or from a user information endpoint.

Implementation:

The API Gateway uses look up tables within a reusable policy to convert roles.

Use Cases

Providing user information

Use Case:

The detailed user information inclusive user roles should be accessible by AWSI services.

Task:

AWSI services can retrieve user information from an user information endpoint.

Implementation:

The API Gateway provides an JWT secured user information endpoint.

Fallback:

In the cases of the necessity of an LDAP interface. The CA directory has to be introduced and integrated into authentication process.

Use Cases

Providing role information

Use Case:

A list of all available AWSI role should be accessible by AWSI services.

Task:

AWSI services can retrieve role information from a role information endpoint.

Implementation:

The API Gateway provides a public accessible role information endpoint.

Use Cases

Issuing of an JSON Web Token

Use Case:

The gateway issues an JSON Web Token for principal authorization at the backend services.

Task:

The API gateway issues an JWT which carries the AWSI roles of the principal for authorization at the backend services. The token has a short life time.

Implementation:

The JSON web token has to be used as an OAuth2 Bearer Token.

Use Cases

Issuing of operational meta information for each request

Use Case:

The gateway issues some meta information to make call sequences visible for the operation team.

Task:

The API gateway attaches a request ID (UUID) for each service request in the HTTP.

Implementation:

An randomly generate UUID is used to set the http header variable X-Request-ID.

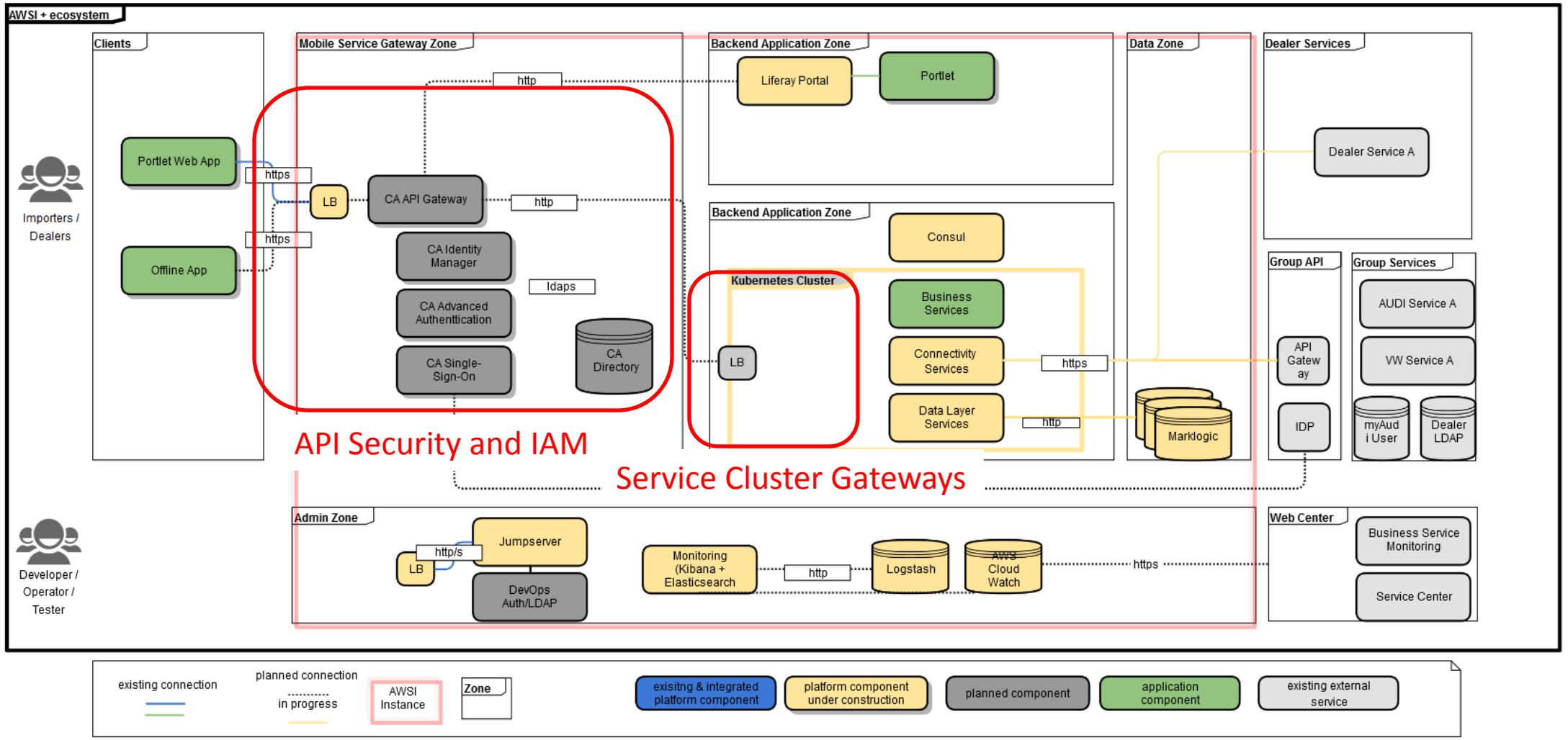
Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- **Architecture**
- Gateway installation and provisioning
- Realization Proposals
- Summary
- Roadmap
- References

Architecture

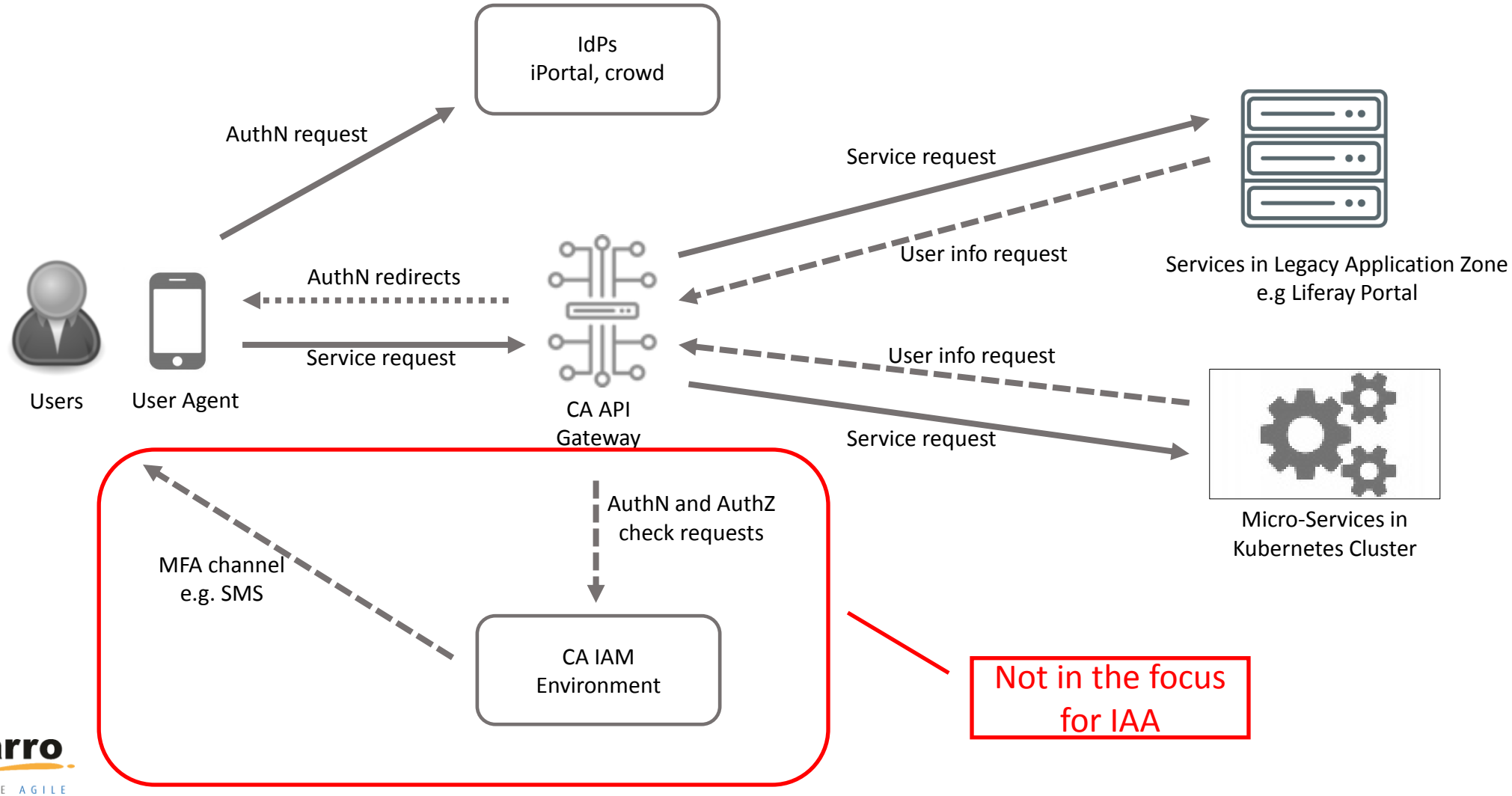
Initial State



nagarro
ENTERPRISE AGILE

Architecture

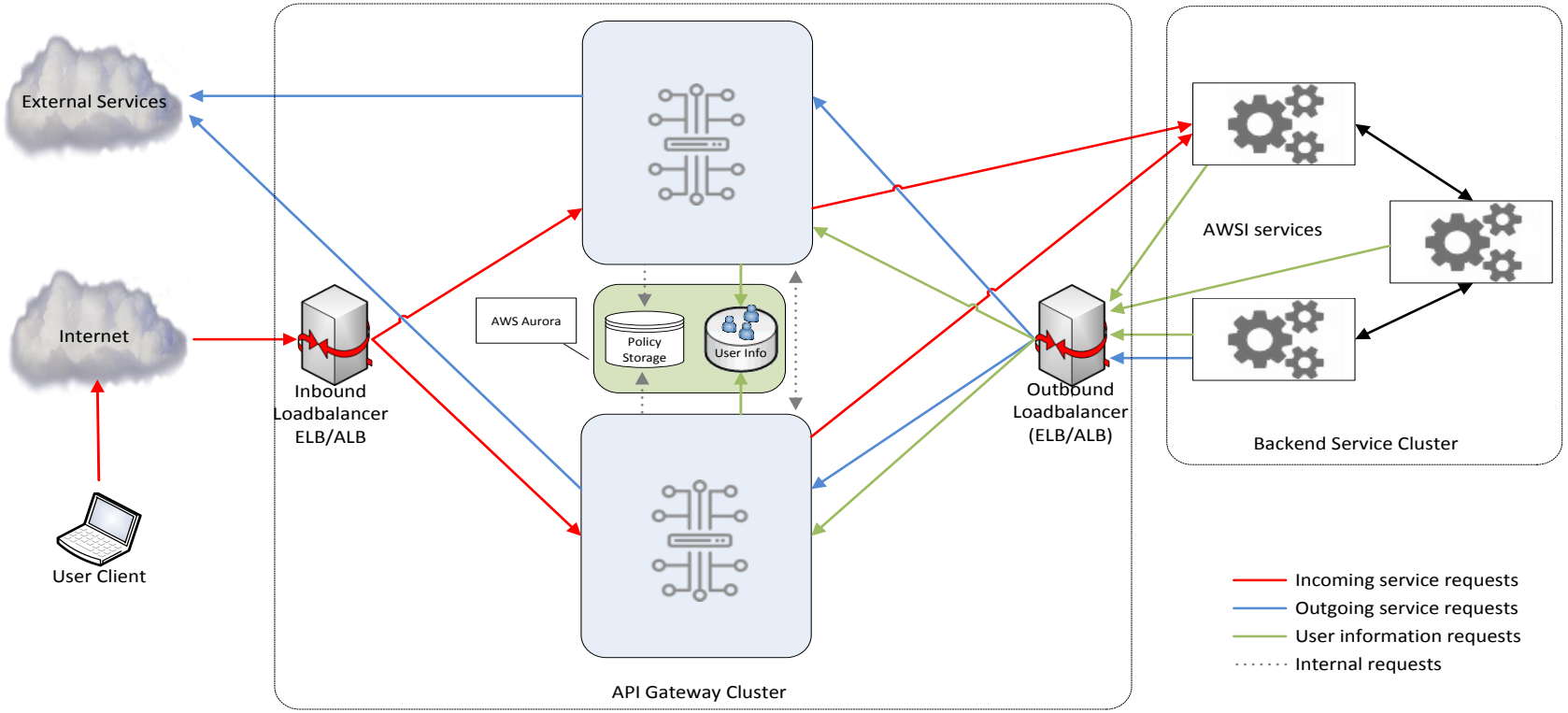
Focus Topic 1 – API Security and IAM



nagarro
ENTERPRISE AGILE

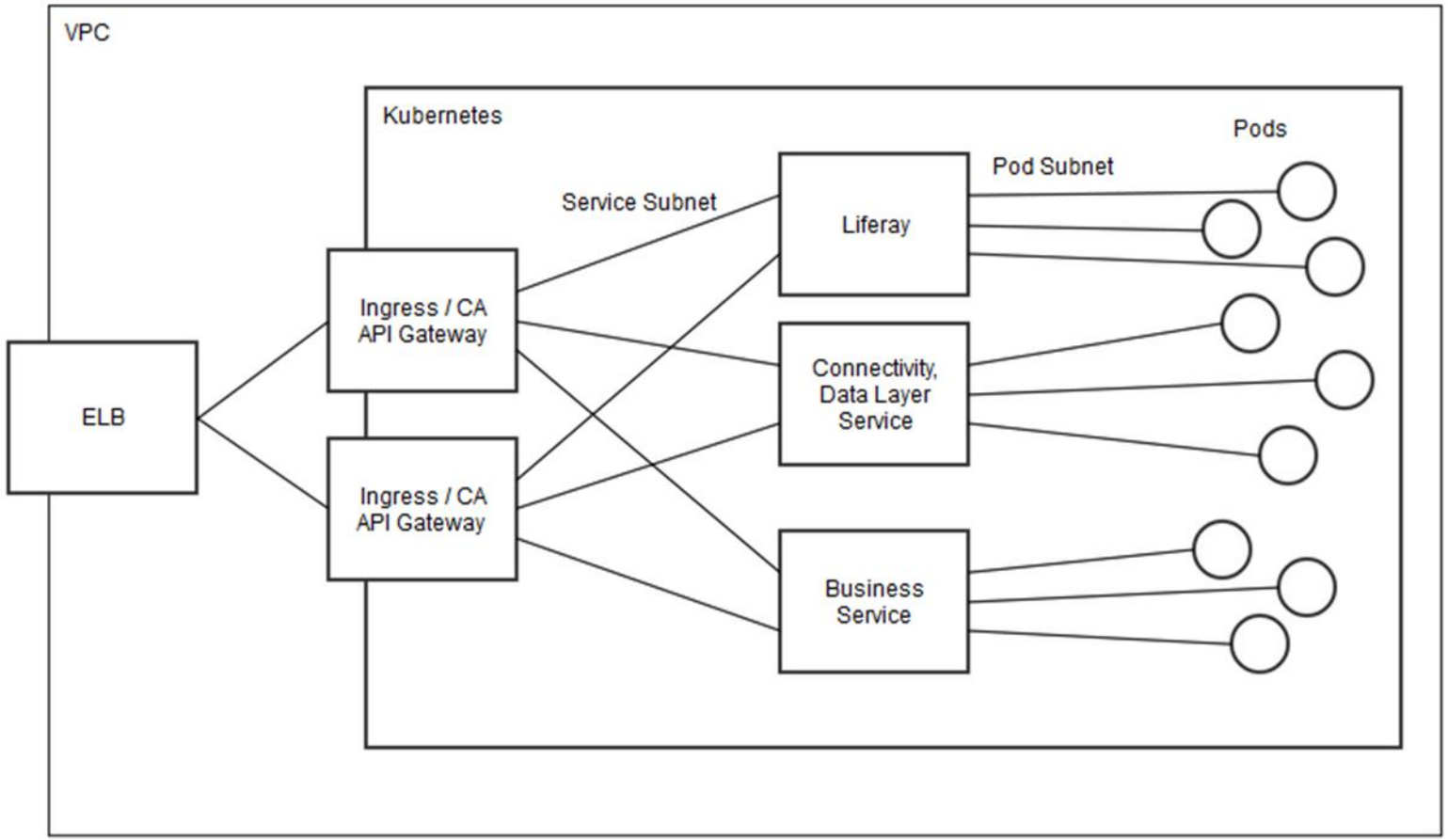
Architecture

Communication flows



Architecture

Focus Topic 2 – Service Cluster Gateways



Targets:

- Investigation of the use of CA API gateways as micro-service front door
- Kubernetes integration
- service access from outside

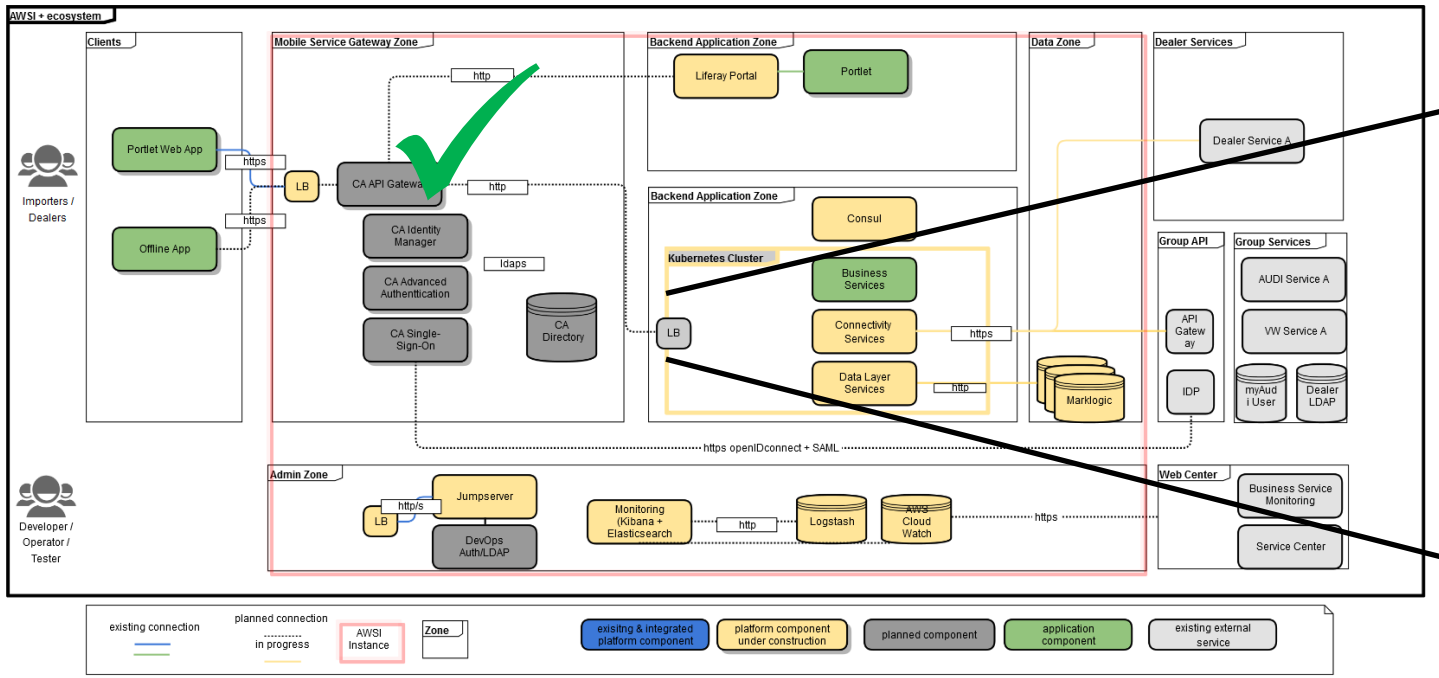
Focus:

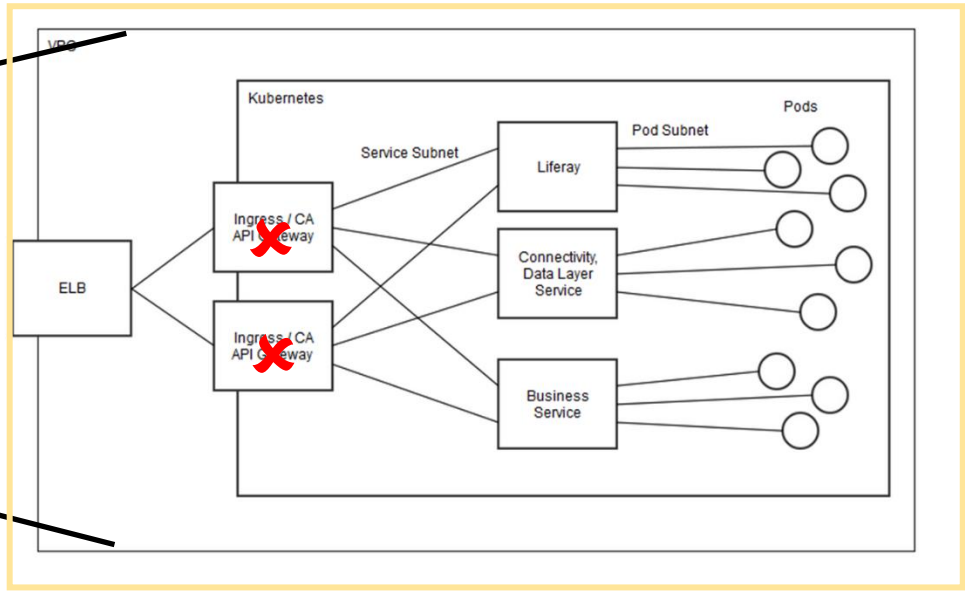
- Dynamic service registration
- Scalability
- Dynamically routing
- Load shedding (limiting requests)
- Service virtualization

Architecture

Placement of API-Gateway

- Our proposal: Restrict API Gateway to one zone only: Mobile Service Gateway Zone
  - (“Group API” not considered here)





Architecture

Placement of API-Gateway

- No real advantages when adding API GW to “Backend zone” / Ingress (Kubernetes)
  - No need for APIs in Ingress – just routing of service-calls
- Some disadvantages we see:
  - More installations necessary
    - ➢ different license model (not user based) might incur higher cost
    - ➢ hardware requirements -> cost
    - ➢ more places where (bad) things can happen
  - Updates / extensions of API GW might not be trivial
    - ➢ better have it in only a few controlled places

- Organisational requirement: API GW should stay under MSG responsibility
- API GW in MSG zone will provide “higher” APIs based on available services

Architecture

Perspective on API Gateway autoscaling

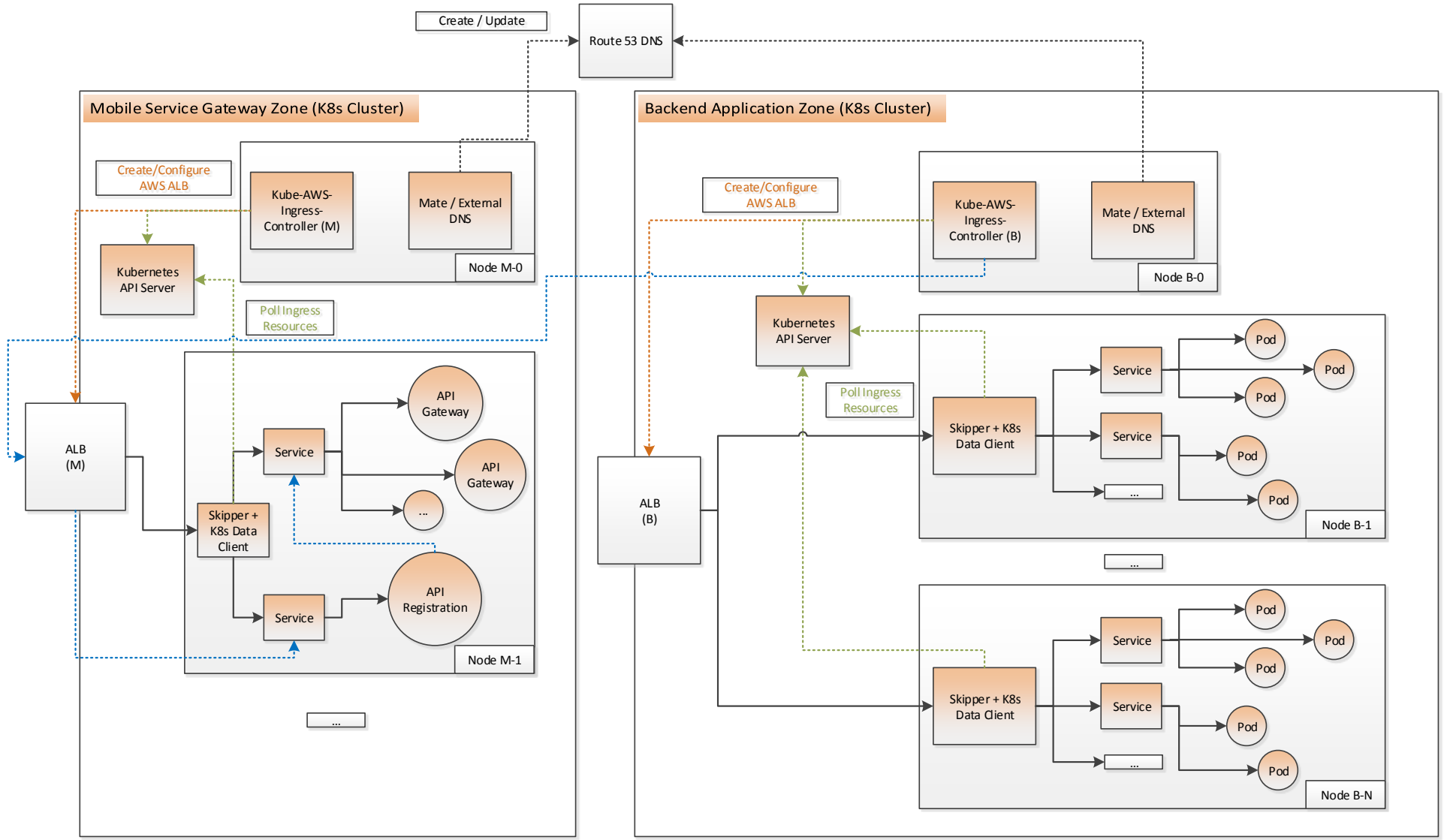
Autoscaling possible for different installation scenarios

- API Gateway in Docker container running under kubernetes control
  - Cloud agnostic approach (public cloud, private Cloud, hybrid, …)
  - Scaling / load balancing covered by kubernetes
    - Cluster Autoscaler on AWS (Glitch: currently no even distribution between AZs)
    - Kubernetes on AWS integrates with ELB

- API Gateway as AMI running as EC2 instances
  - Need trigger for scaling, e.g. CPU / network / memory usage, … (tbd)
  - Simply run in AWS autoscaling group -> scale up / down based on triggers
    - Native integration with ELB/ALB

Architecture

Proposal for target architecture



nagarro
ENTERPRISE AGILE

Architecture

Proposal for target architecture

- Kube-AWS-Ingress-Controller:
  - Deployed as deployment with replicas=1
  - Watch on Ingress resources
  - Configure one Application Load Balancer (ALB/ELBv2)
  - Convenient AWS ASG integration (new nodes are automatically registered in the ALB's Target Group using Skipper NodePort)

- Skipper with Kubernetes data client
  - Deployed as DaemonSet (thus running on every node)
  - Exposes NodePort on every EC2 instance
  - Comes with a Kubernetes data client to automatically update its routes periodically
  - Configure routes defined by host/path matching

- Mate / ExternalDNS: Creates in Route 53 an Alias to the ALB with the name inferred from kubernetes-format or from an annotation

Architecture

Registration process for services (Logical flow)

Always
- service deployment by exposing service to k8s API Server (Ingress)
- registration of service (Skipper with k8s Data-Client listen for new services and register routes automatically)

Externally callable services
- (enhanced) Ingress controller decides (by means of annotation) if service is externally exposed -> Api Registration
- API Registration starts import of service in API-GW

Future
- register service with AUDI registry

nagarro
ENTERPRISE AGILE

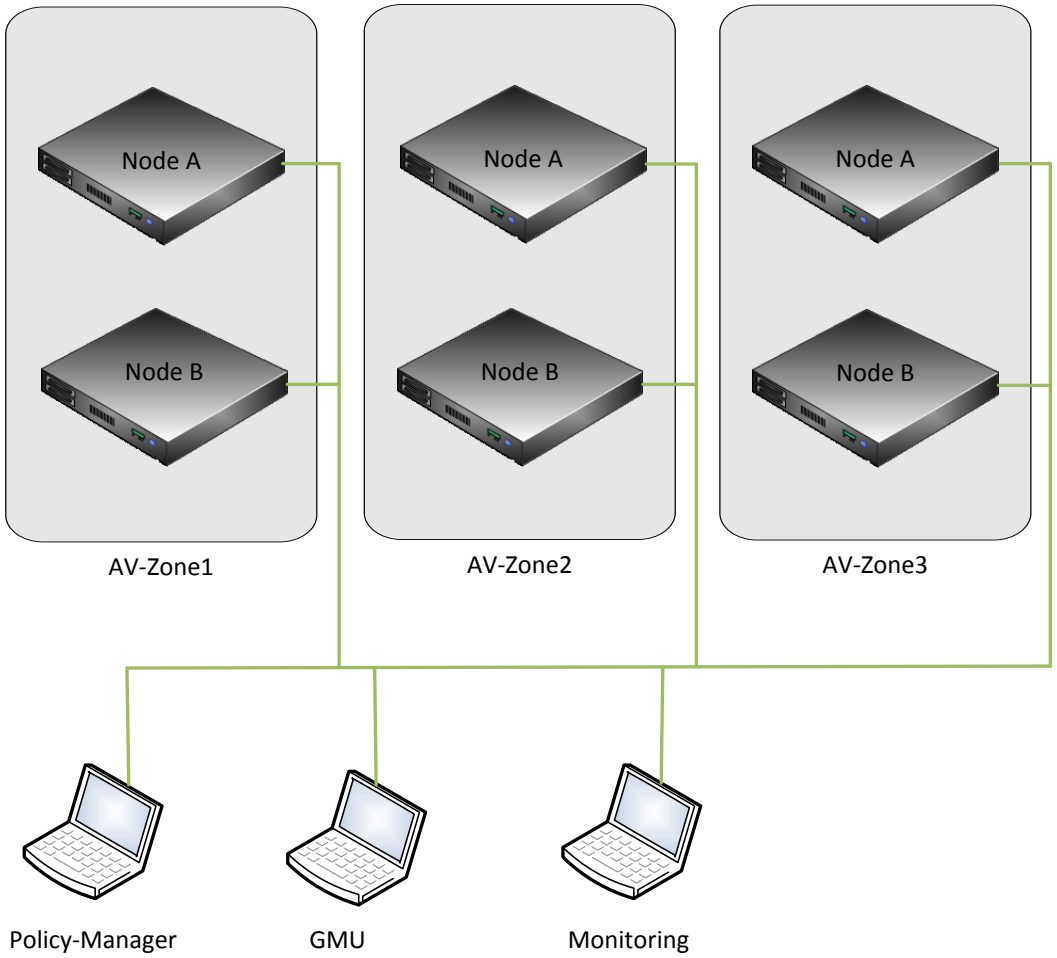
Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- Architecture
- **Gateway installation and provisioning**
- Realization Proposals
- Summary
- Roadmap
- References

API Gateway Installation and Provisioning

Proposal distribution of API Gateway in AWS Availability Zones and Components

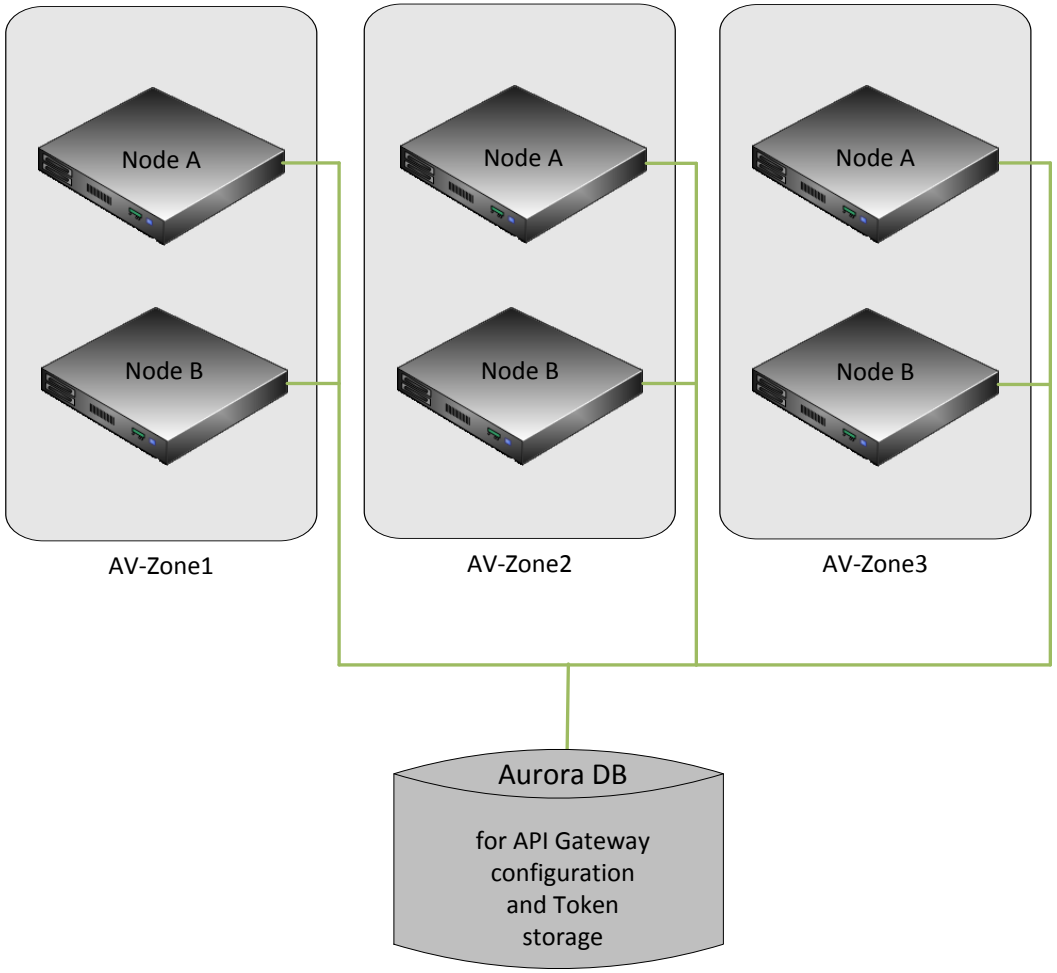


- Service Policies are initially developed in Policy Manager
- Jenkins Job use GMU to retrieve a configuration bundle from Gateway and stores it in a version control repository
- During API Gateway cluster setup, a Jenkins job loads the configuration bundle from the version control repository and stores it on the API Gateways

nagarro
ENTERPRISE AGILE

API Gateway Installation and Provisioning

API Gateway and Persistence



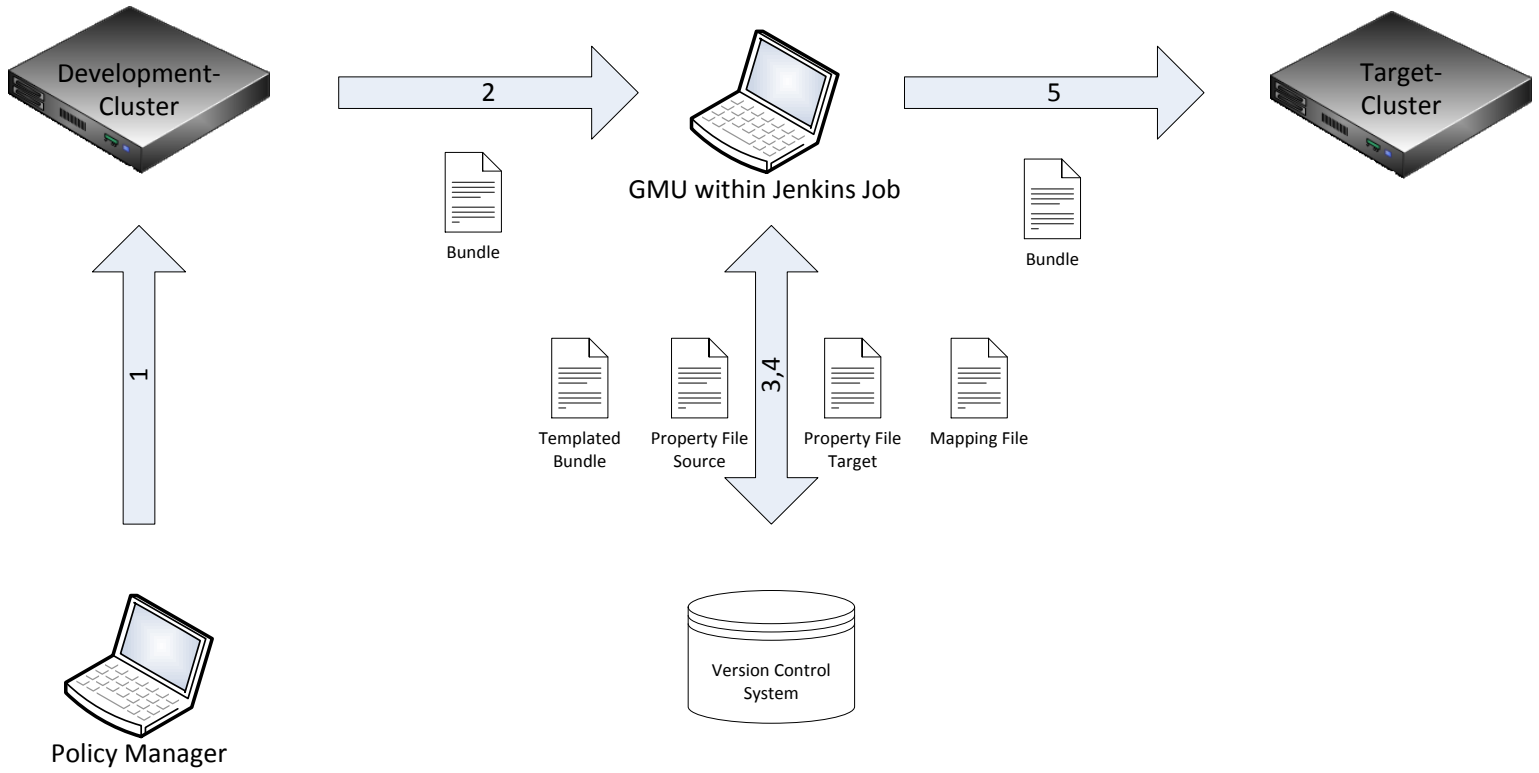
Use of AWS Aurora as:

- API gateway persistence store
- Token and session store
- High available database

nagarro
ENTERPRISE AGILE

API Gateway Installation and Provisioning

Configuration and provisioning process



- Service Policies are initially developed in Policy Manager
- Jenkins Job use GMU to retrieve a configuration bundle from Gateway and stores it in a version control repository
- During API Gateway cluster setup, a Jenkins job loads the configuration bundle from the version control repository and stores it on the API Gateways

Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- Architecture
- Gateway installation and provisioning
- **Realization Proposals**
- Summary
- Roadmap
- References

Realization Proposals

Proposal for an JSON Web Token

JWT Structure:

BASE64URL(UTF8(JWS Protected Header)) || '.' || BASE64URL(JWS Payload) || '.' || BASE64URL(JWS Signature)

Proposal JSON Web Token signing algorithm:

ES256 i.e. ECDSA using P-256 and SHA-256

Realization Proposals

Proposal for JSON Web Token payload

JSON Schema for Token Content

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "AWSI Auth token",
  "description": "Token for AWSI service authentication",
  "type": "object",
  "properties": {
    "iss":{"type": "string",
           "description": "Issuer by default:AWSI API Gateway, see also RFC 7519"},
    "sub":{"type": "string",
           "description": "subject i.e. user ID e.g. Global UserID, Email or id of a client e.g. API key, see also RFC 7519 "},
    "aud": {"type": "string",
            "description": "The audience claim identifies the recipients that the JWT is intended for, see also RFC 7519 "},
    "exp": {"type": "string",
            "description": "The expiration time claim identifies the expiration time on or after which the JWT MUST NOT
                            be accepted for processing, see also RFC 7519 "},
    "iat": {"type": "string",
            "description": The "iat" (issued at) claim identifies the time at which the JWT was issued, see also RFC 7519 "},
    "ver": {"type": "string",
            "description": "version of the token format"},
    "idp": {"type": "string",
            "description": "Idp of user authentication"},
    "roles": {"type": "array",
              "description": "AWSI roles for user or client",
              "items": {"type": "string"}
             }
}
```

nagarro
ENTERPRISE AGILE

Proposals

OAuth response to client application‘s access token request

On success*:

```
{
    "access_token": "<JWT>",
    "token_type": "Bearer",
    "expires_in": "<expiration time in seconds>",
}
```

*refresh_token is for the IAA uses case not supported

Hint: A client application must be able to retrieve the access token from the OAuth authentication response!

On error, all errors messages have with the following structure:

```
{
  “error”:”<error code>”
  “error_description”:”<error description>”
}
```

HTTP-Code	Error-Code	Description
200	n.a.	successful request
400	invalid_request	The request is missing a required parameter, includes an unsupported parameter value (other than grant type), repeats a parameter, includes multiple credentials, utilizes more than one mechanism for authenticating the client, or is otherwise malformed.
400	invalid_grant	The provided authorization grant (e.g., authorization code, resource owner credentials) or refresh token is invalid, expired, revoked and does not match the redirection URI used in the authorization request, or was issued to another client.
401	invalid_client	The apiKey is missing or contains a wrong value. If basic authentication is used or cliend_id and client_secret is passed as parameter, the client_id and/or the client secret can be wrong or missing.
500	unexpected_error	Unspecified error on the server. Only used for not foreseen behavior.

Proposals

Error messages for service requests (OAuth 2 conform)

All error messages are returned as content-type application/json with the following structure:

```
{
  "error":"<error code>"
  "error_description":"<error description>"
}
```

HTTP-Code	Error-Code	Description
200	n.a.	successful request
400	invalid_request	The request is missing a required parameter.
401	invalid_token	There is no or invalid access_token for authentication.
403	invalid_operation	The operation (HTTP verb) may not requested by this URI.
500	unexpected_error	Unspecified error on the server. Only used for not foreseen behavior.

Proposals

Proposal Service versioning on the API Gateway

Best practice:

Put version number of major versions in the URL

Put sub versions numbers (non breaking changes) in the HTTP header

Service URI:

<schema>://<host>[:port]/<service>/<version>/resource

Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- Architecture
- Gateway installation and provisioning
- Realization Proposals
- **Summary**
- Roadmap
- References

Summary

Identity and Access Management

IAM concept:

- Users are authenticated on IdPs outside of AWSI (no user credentials on AWSI)
- Applications (external services) are registered at the Gateway with client credentials
- Backend services are handled as OAuth secured resources (OAuth Service-Provider)
- The OAuth access token (JWT) is passed to the backend services
- Use of OAuth grant types: client credential and saml2-bearer (OAuth-AuthZ-Server)
- Integration of other OAuth/OpenIDConnect IdPs easily possible
- Use of as less as possible proprietary AuthN/AuthZ protocols

Precondition:

- Applications must be able to request and handle an OAuth access token!

Summary

Architecture & Deployment

- Use of CA Docker Images in a Kubernetes cluster
- Cloud agnostic deployments, i.e. runs on different clouds and on premise
- Separate IAM and backend zone -> improved reusability
- Service registration and deployment concept
- High availability
- Self healing capability
- Extendable to further IAM components

Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- Architecture
- Gateway installation and provisioning
- Realization Proposals
- Summary
- **Roadmap**
- References

Proposal Roadmap – AWSI API Layer

Execution Block 1 – Get running

- Infrastructure setup for API Gateways
- Provisioning of 3 static service endpoints and connectivity to backend services
- Enable base policies for services
- Integration of a first IdP federation
- Design and establish JWT (JWS/JWE) for service access*
- Design and implement User info endpoint (OpenId Connect compatible)*
- Mockup missing interfaces
- Establish API Gateway provisioning process via Gateway Migration Utility*
- Design Service registration process*
- Collect further IAM use cases

* Reuse in General API Layer

Proposal Roadmap – AWSI API Layer

Execution Block 2 – Completion of IAM

- Infrastructure setup IAM components (after IAA)
- Integration of further IdPs
- Establish user registration process* (after IAA)
- Define Webservice Interface for Identity Manager (after IA)
- Depend from IAA use case:
  - Implementation of the service IDP relation mapping, iPortal Use Case i.e. redirect to IdPs
  - Design general interface for Authorization provisioning for Portals (AuthZ/role concept on GW) or rough service authorization (DDOs prevention)

* Reuse in General API Layer

Proposal Roadmap – AWSI API Layer

Execution Block 3 – Prepare Operation

- Infrastructure setup of all CA components for High Availability*
- Register CA components for central monitoring, logging and auditing
- Prepare products update/upgrade process
- Security hardening where needed (OS, Cipher Suites, etc.)

* Partial reuse in General API Layer

Proposal Roadmap – AWSI API Layer

Execution Block 4 – Continuous Optimization

- Integration of new IdPs
- Creation of new federations
- Improve automatic service provisioning*
- Optimize deployments e.g. move to Docker and Kubernetes for further components*
- Adaptions for cost optimizations on AWS*
- Enable step up Authentication
- Establish Inter-Region communication

* Reuse in General API Layer

Agenda

API Management Proposal for AWSI

- Focus
- Requirements
- Use Cases
- Architecture
- Gateway installation and provisioning
- Realization Proposals
- Summary
- Roadmap
- **References**

References

Relevant RFCs

Reference	URL
OAuth 2.0	https://tools.ietf.org/html/rfc6749
OAuth Bearer Token	https://tools.ietf.org/html/rfc6750
Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client AuthN and AuthZ Grants	https://art.tools.ietf.org/html/rfc7522
JSON Web Token (JWT) Profil for OAuth 2.0 Client AuthN and AuthZ Grants	https://art.tools.ietf.org/html/rfc7523
SAML core	http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
SAML bindings	http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
JSON Web Signature (JWS)	https://datatracker.ietf.org/doc/html/rfc7515
JWS signature algorithm	http://www.rfc-editor.org/info/rfc7518

Architecture

Focus Topic 1 – CA IAM Environment