NIST

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

CSRC

3rd Open Security Controls Assessment Language (OSCAL) Workshop

Kubernetes Policy WG

kubernetes

CLOUD NATIVE COMPUTING FOUNDATION

Anca Sailer, Jim Bugwadia, Jayashree Ramanat, Robert Ficcaglia

Kubernetes Policy Working Group (WG)

Provide an overall architecture that describes both the current policy related implementations and future policy proposals in Kubernetes. Through a collaborative method, we want to present both operators and users a universal view of policy architecture in Kubernetes.

GitHub:
kubernetes-sigs/wg-policy-prototypes

Slack:
https://slack.k8s.io/#wg-policy

Open Meetings
Wed 8:00 AM Pacific/ 11 AM Eastern, Every two weeks

Current Projects

1. Policy Report Custom Resource Definition (CRD)
2. OSCAL-aligned Policy Report
3. Kubernetes Policy Management Whitepaper

Policy Report
Custom Resource Definition (CRD)

Motivations

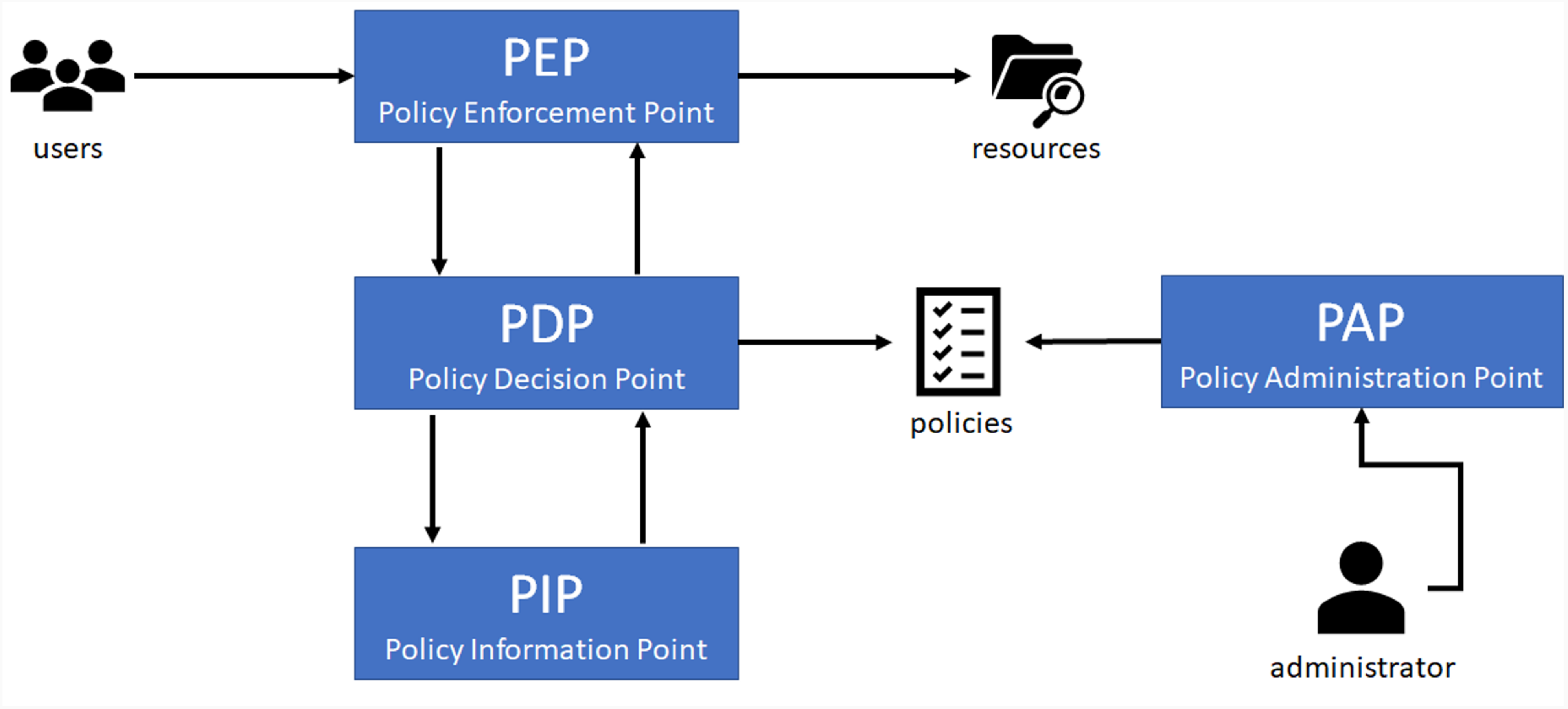
- How to standardize “policy interface”, similar to CSI, CNI, etc.
- Hard to formalize as “policy” covers several different areas of concerns e.g. images, runtime, configuration, cluster etc.
- Difficult to standardize a policy language

If not a formal policy interface, what portion of the policy life-cycle would be impactful to standardize across domains and use cases?

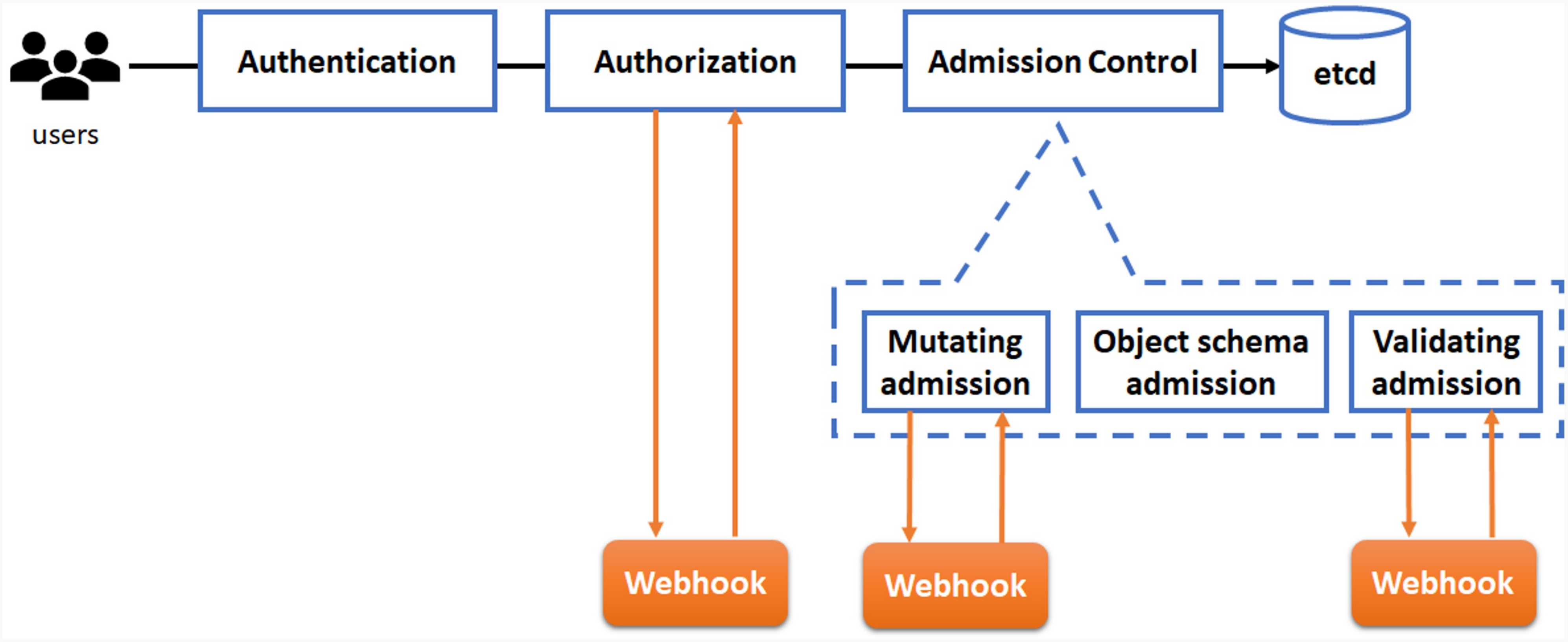
Policy Report

- Kubernetes resource (namespaced or cluster-wide)
- Definition only - controller not included
- Focused on current data. Historical data to be managed externally.
- Flexible reporting options for different engines
- Works with all K8s machinery and tools
- **Align with industry and public sector efforts (e.g. OSCAL)**

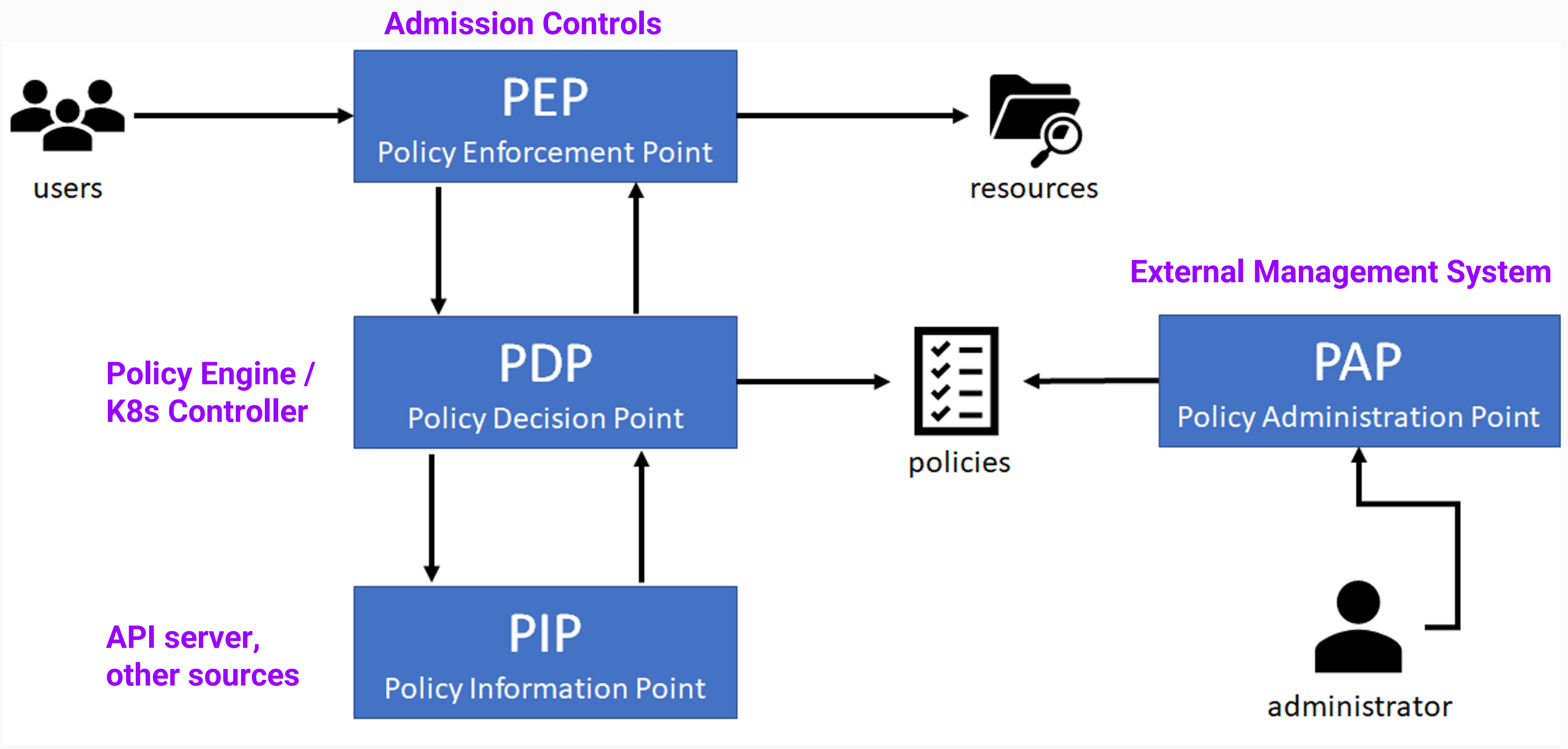
XACML (eXtensible Access Control Markup Language) Reference Architecture



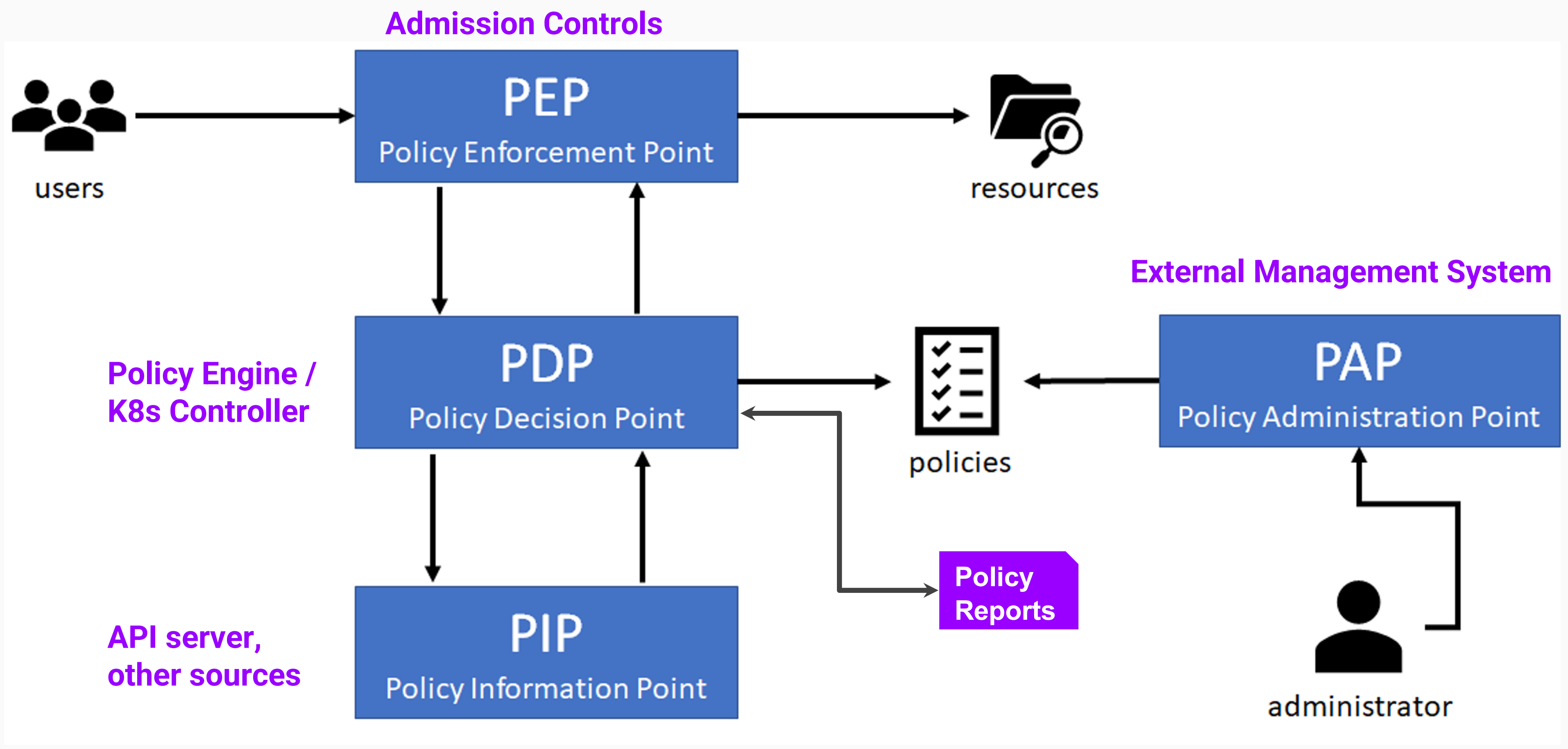
Kubernetes API Request Flow



XACML Reference Architecture → Kubernetes



XACML Reference Architecture → Kubernetes with Policy Report



Policy Report Adoption

Tool	Area of concern	Status
Kyverno	Configuration Security	Completed
Policy Reporter	UI / Reporting / Notifications	Completed
kube-bench	CIS Kubernetes Benchmarks (Control plane, worker nodes)	Completed
Falco	Runtime Security	Completed
Trivy	Vulnerability scanning	Completed
KubeArmor	Runtime Security	Completed

OSCAL Policy Report

MAPPING

All Policy Result attributes have been designed with mapping to OSCAL in mind.

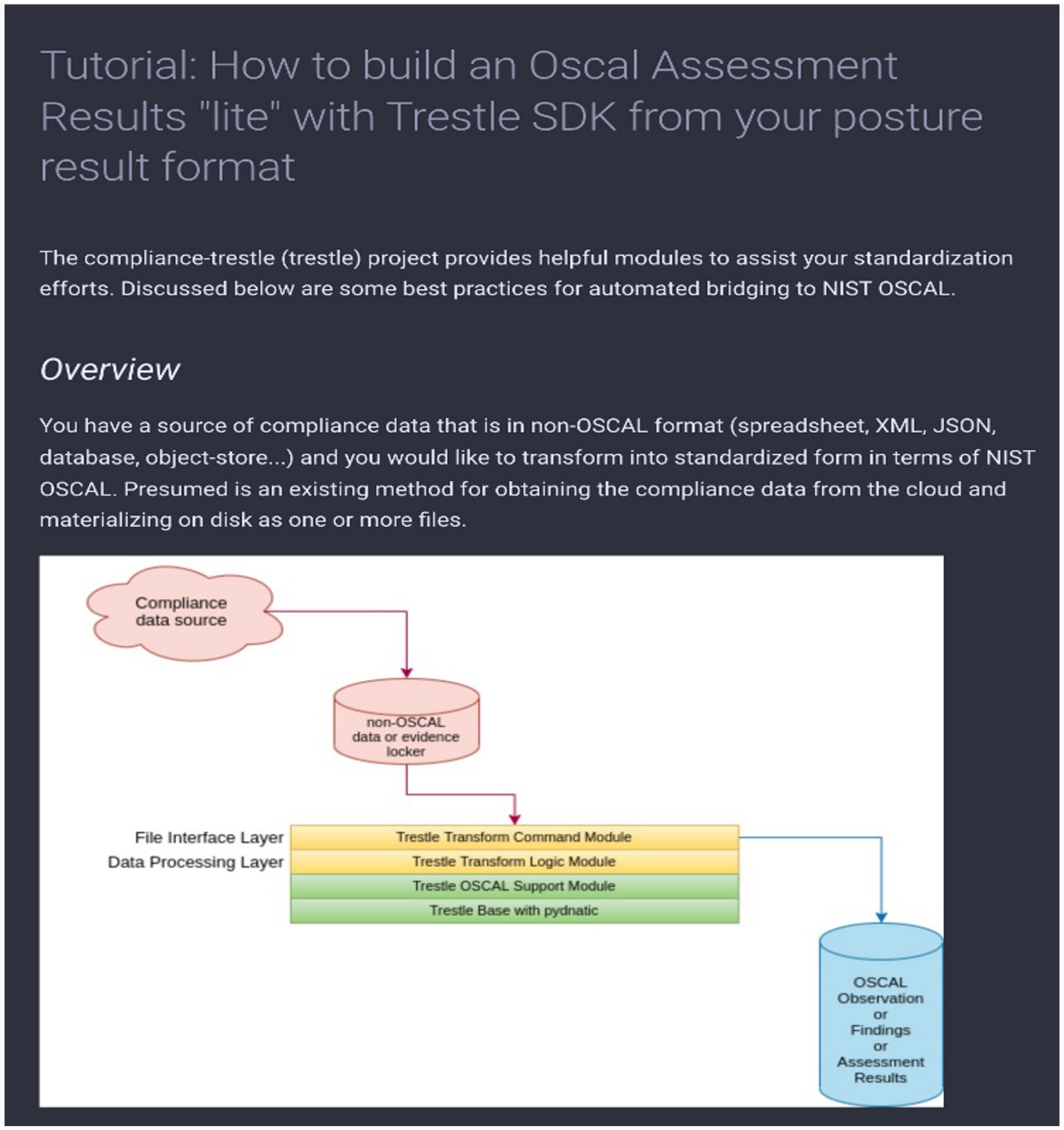
Initial mapping is focused on OSCAL Observations.

OSCAL json Path	property name	property class	yaml path
local-definitions.inventory-items.prop	scope.apiVersion		scope.apiVersion
local-definitions.inventory-items.prop	scope.kind		scope.kind
local-definitions.inventory-items.prop	scope.name		scope.name
local-definitions.inventory-items.prop	scope.namespace	scc_scope	scope.namespace
result.title	N/A		metadata.name
result.description	N/A		metadata.labels.wgpolicyk8s.io/engine
result.description	N/A		metadata.labels.policy.kubernetes.io/engine
result.prop	apiVersion		apiVersion
result.prop	kind		kind
result.prop	metadata.namespace		metadata.namespace
result.prop	metadata.annotations.name		metadata.annotations.name
result.prop	metadata.annotations.category		metadata.annotations.category
result.prop	metadata.annotations.file		metadata.annotations.file
result.prop	metadata.annotations.version		metadata.annotations.version
result.prop	results.policy	scc_rule	results.policy
result.observation.prop	results.message	scc_description	results.message
result.observation.prop	results.result	scc_result	results.result

trestle

Trestle OSCAL object model can easily be used to convert content:

- Excel files: https://github.com/IBM/compliance-trestle-demos/tree/develop/CIS_controls
- XML content: https://github.com/IBM/compliance-trestle-demos/tree/develop/ISM_catalog_profile



Tutorial: How to build an Oscal Assessment Results "lite" with Trestle SDK from your posture result format

The compliance-trestle (trestle) project provides helpful modules to assist your standardization efforts. Discussed below are some best practices for automated bridging to NIST OSCAL.

Overview

You have a source of compliance data that is in non-OSCAL format (spreadsheet, XML, JSON, database, object-store...) and you would like to transform into standardized form in terms of NIST OSCAL. Presumed is an existing method for obtaining the compliance data from the cloud and materializing on disk as one or more files.

Compliance data source
non-OSCAL data or evidence locker
File Interface Layer
Data Processing Layer
Trestle Transform Command Module
Trestle Transform Logic Module
Trestle OSCAL Support Module
Trestle Base with pydnatic
OSCAL Observation or Findings or Assessment Results

Thanks!

GitHub

[Kubernetes Policy Working Group]

[Policy Prototypes Repo]