Red Hat Confidential - Associates only

AI-Powered Performance Insights

Integrating OVS/OVN automatic performance regression analysis with LLMs

Joe Talerico, Mohit Sheth, Raul Sevilla, José Castillo

Perf&Scale Engineering

whoarewe

- 13 years in the Performance Team
- Worked on Virt, SPEC, OpenStack, and OpenShift

Joe
Talerico

JOSE
CASTILLO LEMA

- 6 years in Red Hat
- 4 years in perf/scale department
- 2 years as Telco Cloud Architect in the Solutions & Technology Practices team

- 10 years in Red Hat
- 6 years in the OCP Perf&Scale Team
- 4 years in the Iberia Services Team

Raúl
Sevilla

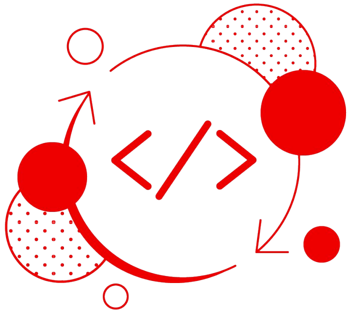
Agenda

1. The problem
2. Shift-left engineering
3. Continuous Performance Testing (CPT)
4. Workflow
5. Tooling
6. How does AI fit in?
7. Challenges and next steps

The problem

- Performance testing often occurs **too late in the pipeline**
- Late detection of performance issues increases **cost**
- Real-world consequences: downtime, latency, etc.

Shift-left engineering



Development

Shift left

Performance Engineering

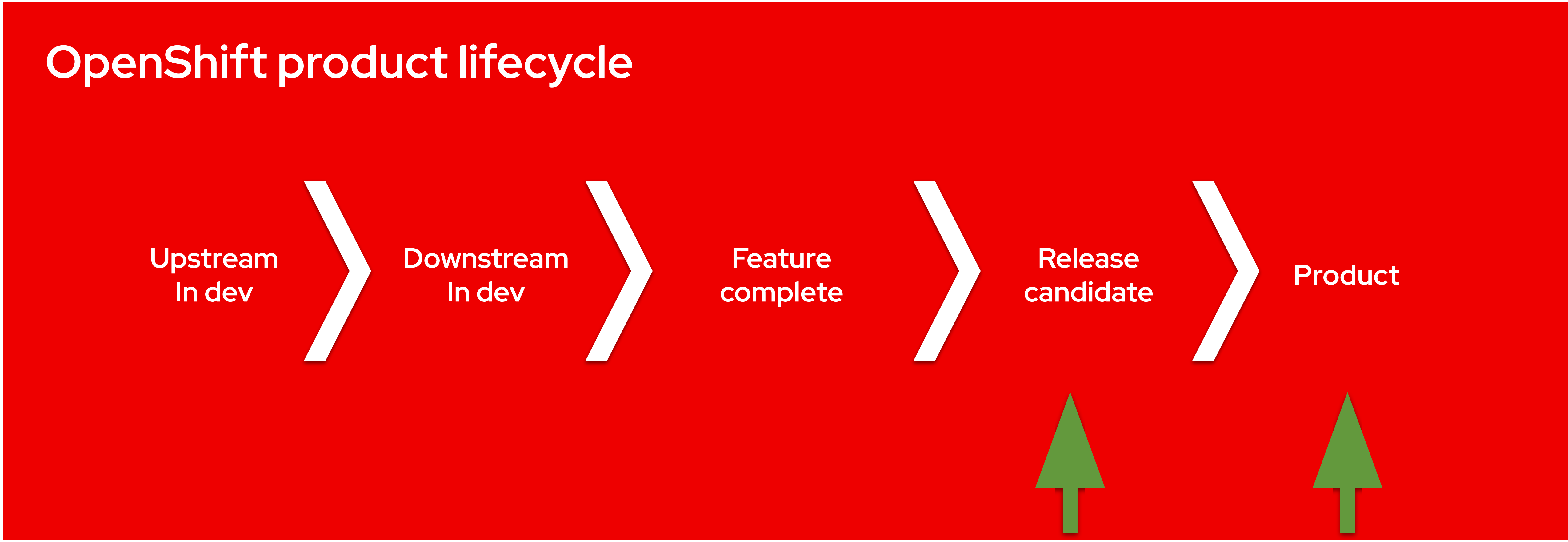
- Testing
- Tuning
- Capacity planning

Shift right

Operations

- Monitoring
- Capacity planning

Performance Testing – The past



OpenShift product lifecycle

Upstream In dev

Downstream In dev

Feature complete

Release candidate

Product

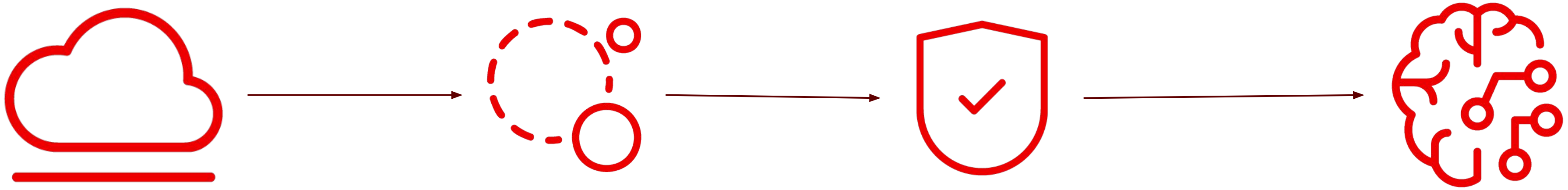
At best we normally would fit in at the end of the development cycle

What is CPT?

Continuous Performance Testing (CPT) is the practice of executing performance benchmarks (tests) continuously throughout the development lifecycle.

The execution of said Performance benchmarks is **orchestrated in an automated pipeline** (like Jenkins, Prow, etc.), preferably in the pipeline your developers are working out of.

Workflow



Deploy product

Run perf test

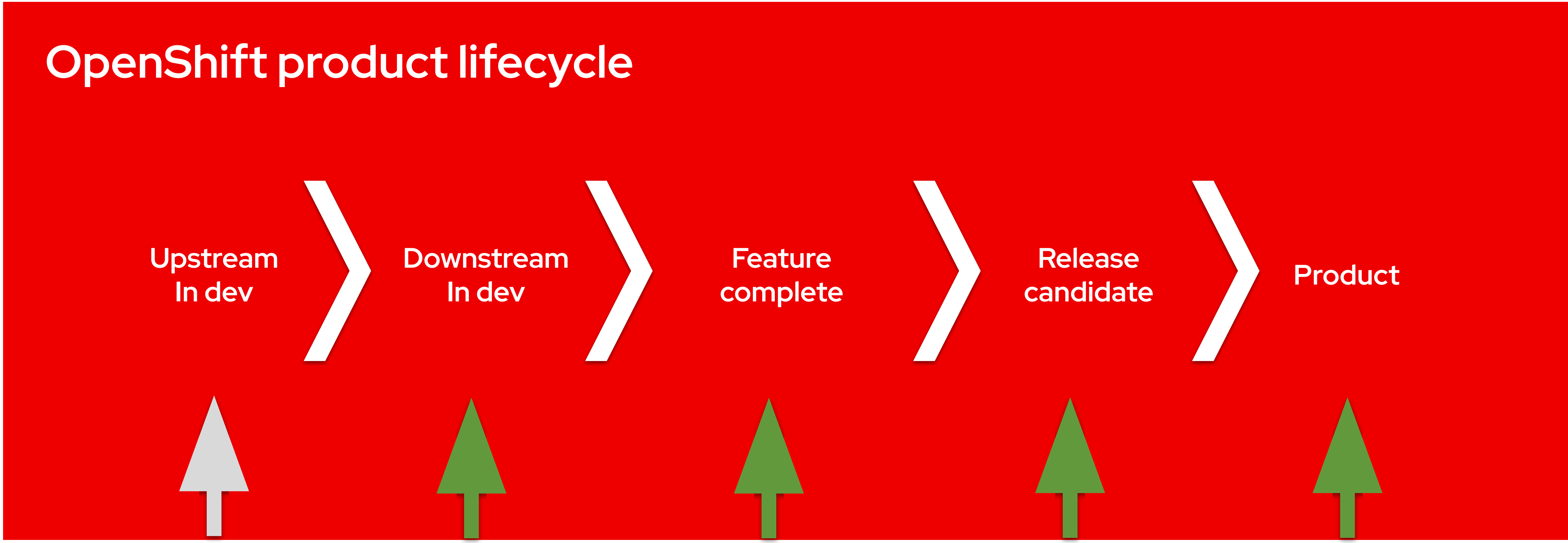
Store results

Perf regression analysis

Why CPT?

- Shifts Performance testing left
- Earlier feedback enables faster fixes
- Reduces overall testing costs
- Improves developer ownership of performance → Continuous Improvement Mindset
- More time for technical improvements
- Allows us to drastically improve the coverage matrix and frequency of the testing
  - More data → Need for automatic **performance regression frameworks**
    - **Change point detection** mechanisms

CPT - Goals



OpenShift product lifecycle

Upstream In dev

Downstream In dev

Feature complete

Release candidate

Product

After adopting CPT, where we fit into the lifecycle.

We would like to see how we can enable the same process upstream!

Cultural shift

- From siloed to **dev + Performance collaboration**
- Embed performance mindset in development sprints
- Performance as part of 'Definition of Done'

Key principles to **navigate change**:

- Document workflows and KPI’s
- Collaboration with QE counterparts is key
  - Handoffs
  - Responsibility matrix

PerfScale x Openshift Networking

2025 in review

- Net New Workloads added (Control & Data-path)
  - UDN L2 & L3
  - Virt UDN
  - Networkpolicy
  - Egress IP
  - BGP
- 30+ downstream bugs opened
- Improved tooling to identify regressions in a PR
- BiWeekly engagement with developers

Achievements

- Catch performance/scalability regressions in an early stage
- Redefined a new test/platform coverage matrix standard

5 platforms

AWS, Azure, GCP, IBM, Baremetal +
Managed services offerings: ROSA and ARO

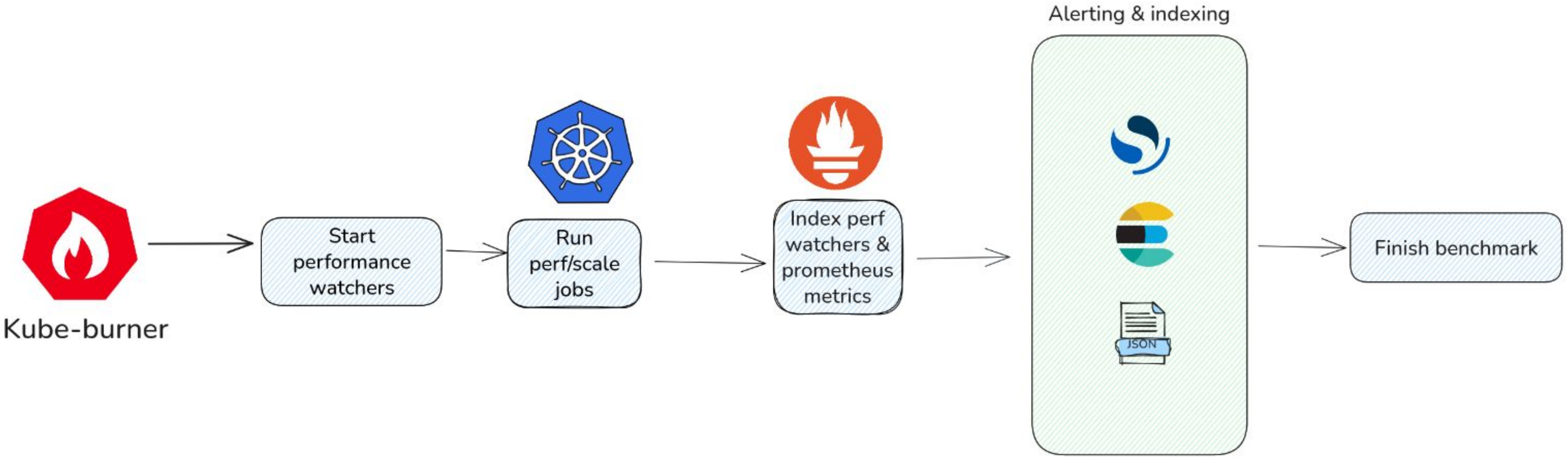
+100 weekly tests

Covering nightly, early candidates, release candidates, stable releases and long term support

Platform +

- OpenShift Virtualization
- Layered products (OLS, ACS, Kepler, …)

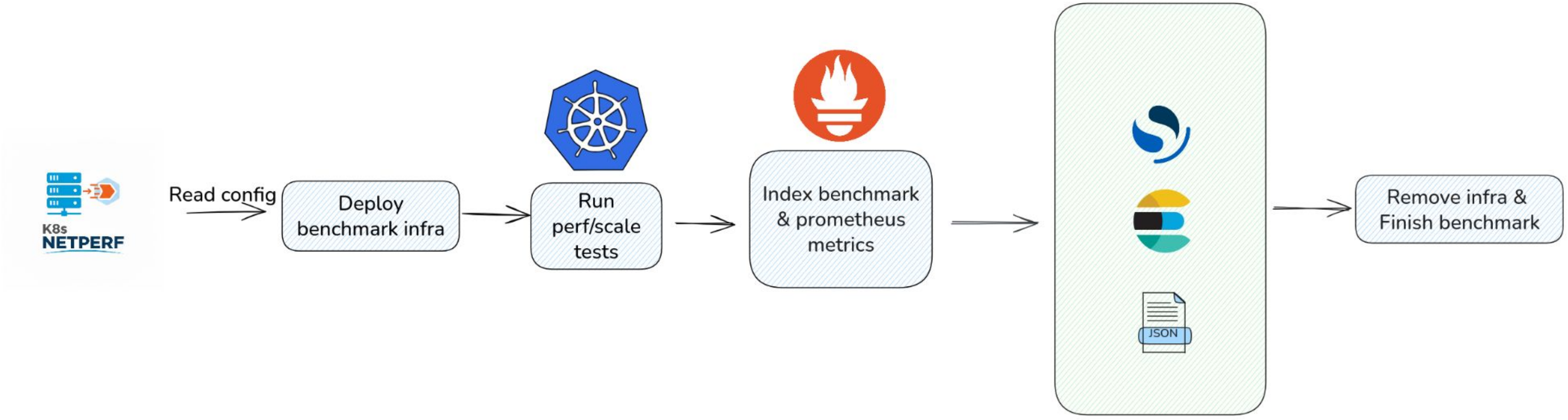
Control-Plane Tooling



Supported indexers:
- OpenSearch
- Elasticsearch
- Local file

Data-Plane Tooling - k8s-netperf

Red Hat Confidential - Associates only



K8s NETPERF

Read config

Deploy benchmark infra

Run perf/scale tests

Index benchmark & prometheus metrics

JSON

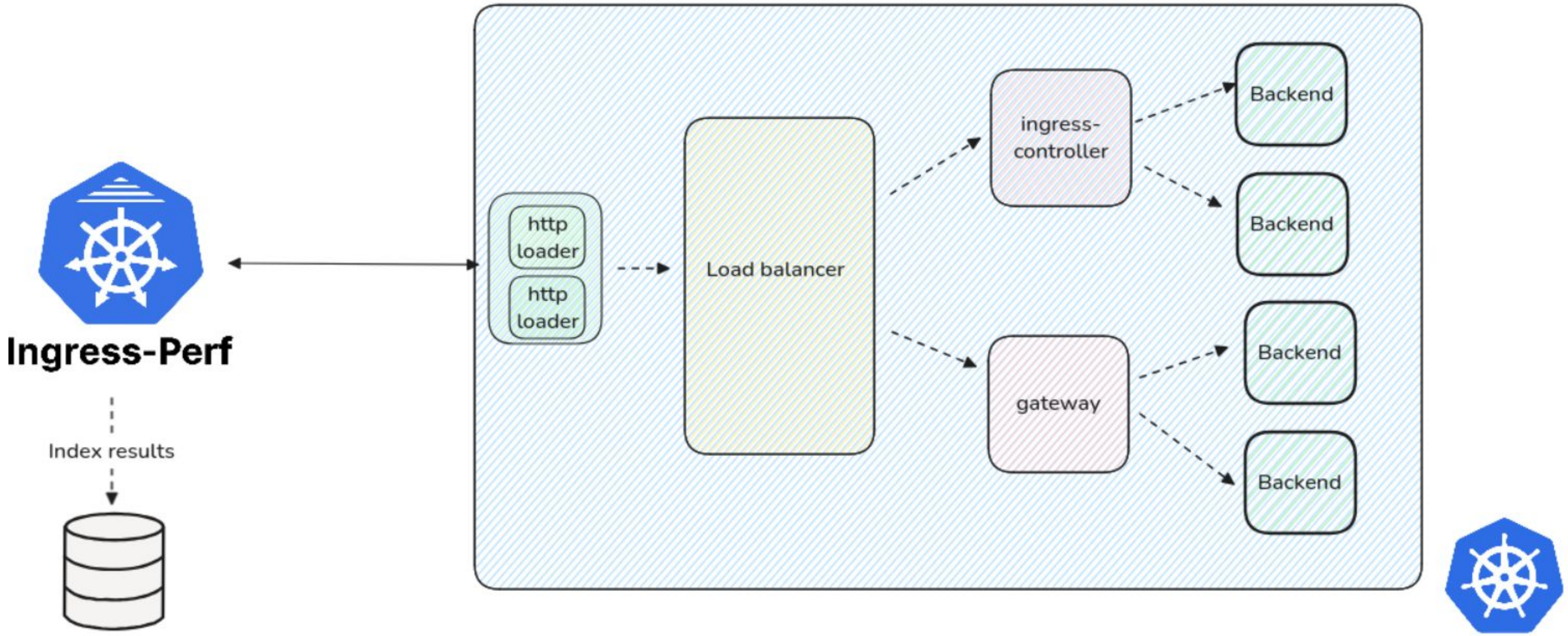
Remove infra & Finish benchmark

Supported indexers:
- OpenSearch
- Elasticsearch
- Local file

k8s-netperf:
- Deploys benchmark assets in the given cluster
- Drives benchmark orchestration and results collection
- Normalizes and index benchmark results along with Prometheus metrics in local files or ElasticSearch/OpenSearch
- Garbage collects benchmark assets

Supports throughput and latency benchmarks:
- Pod 2 pod via SDN or hostNetwork (including VMs)
- Pod 2 service
- Multiple drivers: netperf, uperf, iperf3.

Data-Plane Tooling - ingress-perf

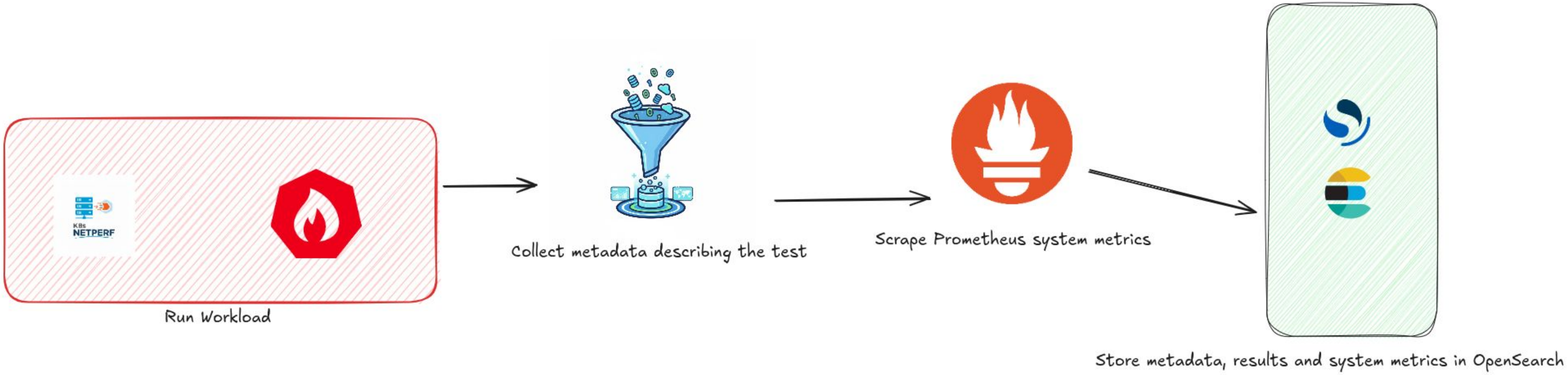


Ingress-perf:

- Deploys benchmark assets in the given cluster
- Drives benchmark orchestration and results collection
- Normalizes and index benchmark results along with Prometheus metrics in local files or ElasticSearch/OpenSearch
- Garbage collects benchmark assets

Regression Analysis - Tooling

Across all of our Performance and Scale Tooling, there is a common workflow.



Run Workload

Collect metadata describing the test

Scrape Prometheus system metrics

Store metadata, results and system metrics in OpenSearch

With this common workflow and data warehouse, we can use a common tool for the regression analysis, called Orion

Orion works by querying the data warehouse for all results that *match a specific fingerprint*. Once we have all the matches, Orion will then collect the desired system metrics the user provided and run the provided regression analysis algorithm against the dataset – We use Apache Otava (e-divisive means algorithm, learn more)

Regression Analysis - Tooling

Example of an Orion configuration:

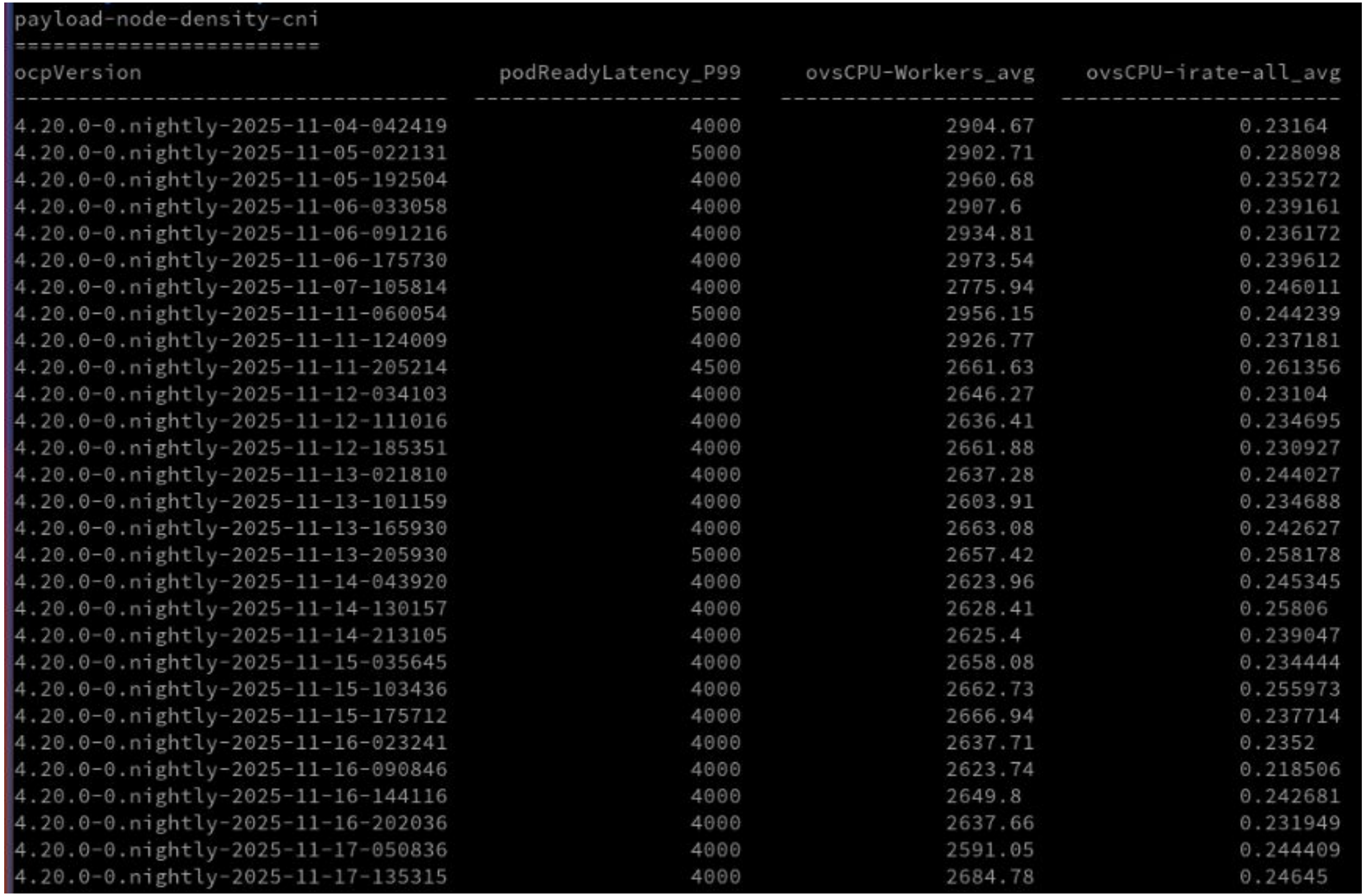
```
tests :
  - name : 24-node-scale-cdv2
    index: {{ es_metadata_index }}
    benchmarkIndex: {{ es_benchmark_index }}
    metadata:
      platform: AWS
      clusterType: self-managed
      masterNodesType: m6a.xlarge
      masterNodesCount: 3
      workerNodesType: m6a.xlarge
      workerNodesCount: 24
      benchmark.keyword: cluster-density-v2
      ocpVersion: {{ version }}
      networkType: OVNKubernetes
      jobType: {{ jobtype | default('periodic') }}
      not:
        stream: okd
```

This example fingerprint will retrieve:

- all runs on AWS
- `m6a.xlarge` control-plane
- `m6a.xlarge` workers
- 24 workers
- The benchmark was cluster-density-v2

Regression Analysis - Tooling

Example of an Orion OVN analysis:



```
payload-node-density-cni
========================
ocpVersion                           podReadyLatency_P99    ovsCPU-Workers_avg    ovsCPU-irate-all_avg
----------------------------------  ---------------------   --------------------  ----------------------
4.20.0-0.nightly-2025-11-04-042419                   4000                2904.67                0.23164
4.20.0-0.nightly-2025-11-05-022131                   5000                2902.71                0.228098
4.20.0-0.nightly-2025-11-05-192504                   4000                2960.68                0.235272
4.20.0-0.nightly-2025-11-06-033058                   4000                2907.6                 0.239161
4.20.0-0.nightly-2025-11-06-091216                   4000                2934.81                0.236172
4.20.0-0.nightly-2025-11-06-175730                   4000                2973.54                0.239612
4.20.0-0.nightly-2025-11-07-105814                   4000                2775.94                0.246011
4.20.0-0.nightly-2025-11-11-060054                   5000                2956.15                0.244239
4.20.0-0.nightly-2025-11-11-124009                   4000                2926.77                0.237181
4.20.0-0.nightly-2025-11-11-205214                   4500                2661.63                0.261356
4.20.0-0.nightly-2025-11-12-034103                   4000                2646.27                0.23104
4.20.0-0.nightly-2025-11-12-111016                   4000                2636.41                0.234695
4.20.0-0.nightly-2025-11-12-185351                   4000                2661.88                0.230927
4.20.0-0.nightly-2025-11-13-021810                   4000                2637.28                0.244027
4.20.0-0.nightly-2025-11-13-101159                   4000                2603.91                0.234688
4.20.0-0.nightly-2025-11-13-165930                   4000                2663.08                0.242627
4.20.0-0.nightly-2025-11-13-205930                   5000                2657.42                0.258178
4.20.0-0.nightly-2025-11-14-043920                   4000                2623.96                0.245345
4.20.0-0.nightly-2025-11-14-130157                   4000                2628.41                0.25806
4.20.0-0.nightly-2025-11-14-213105                   4000                2625.4                 0.239047
4.20.0-0.nightly-2025-11-15-035645                   4000                2658.08                0.234444
4.20.0-0.nightly-2025-11-15-103436                   4000                2662.73                0.255973
4.20.0-0.nightly-2025-11-15-175712                   4000                2666.94                0.237714
4.20.0-0.nightly-2025-11-16-023241                   4000                2637.71                0.2352
4.20.0-0.nightly-2025-11-16-090846                   4000                2623.74                0.218506
4.20.0-0.nightly-2025-11-16-144116                   4000                2649.8                 0.242681
4.20.0-0.nightly-2025-11-16-202036                   4000                2637.66                0.231949
4.20.0-0.nightly-2025-11-17-050836                   4000                2591.05                0.244409
4.20.0-0.nightly-2025-11-17-135315                   4000                2684.78                0.24645
```

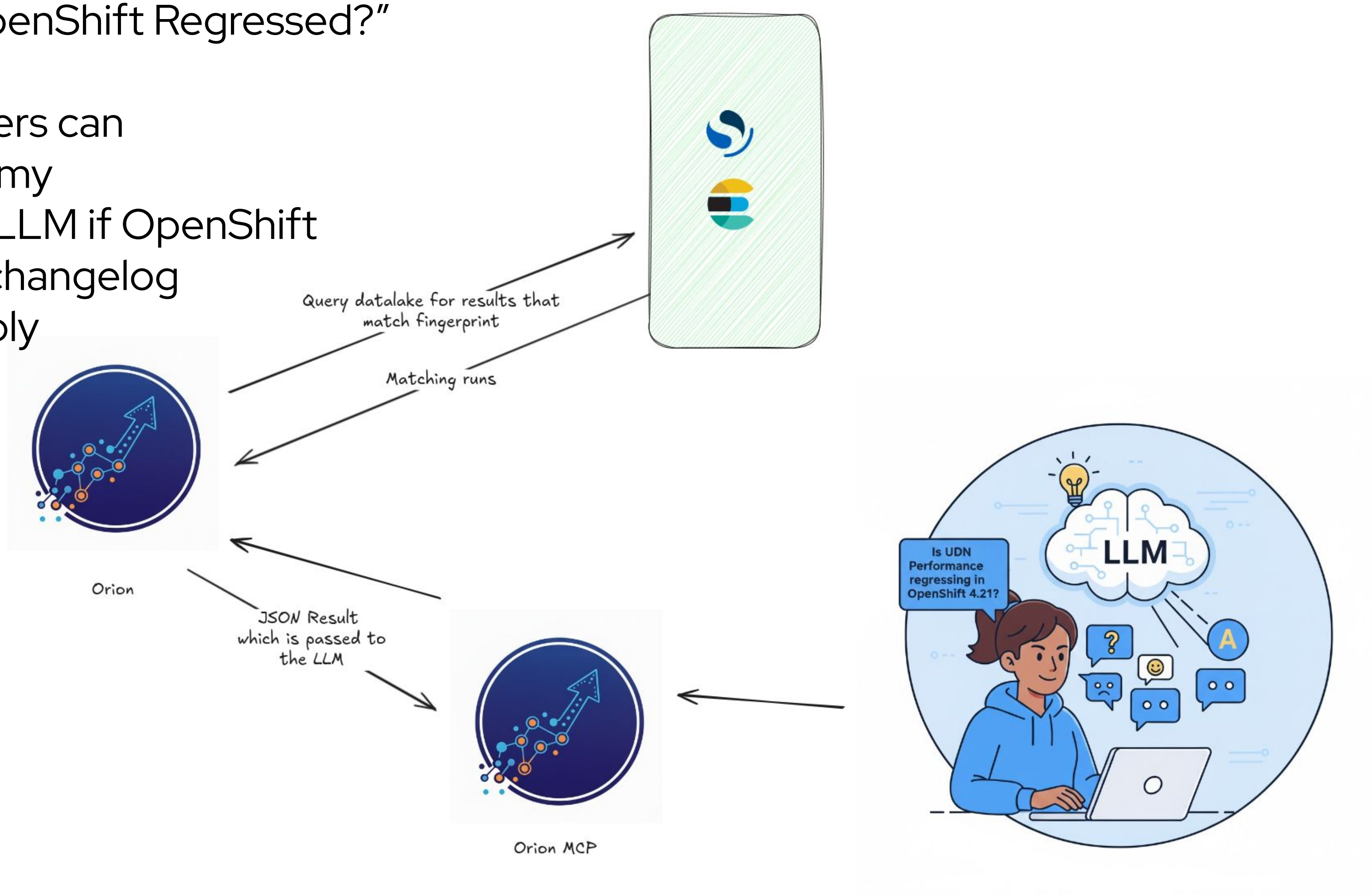
*Cropped due to size

Where is the AI?

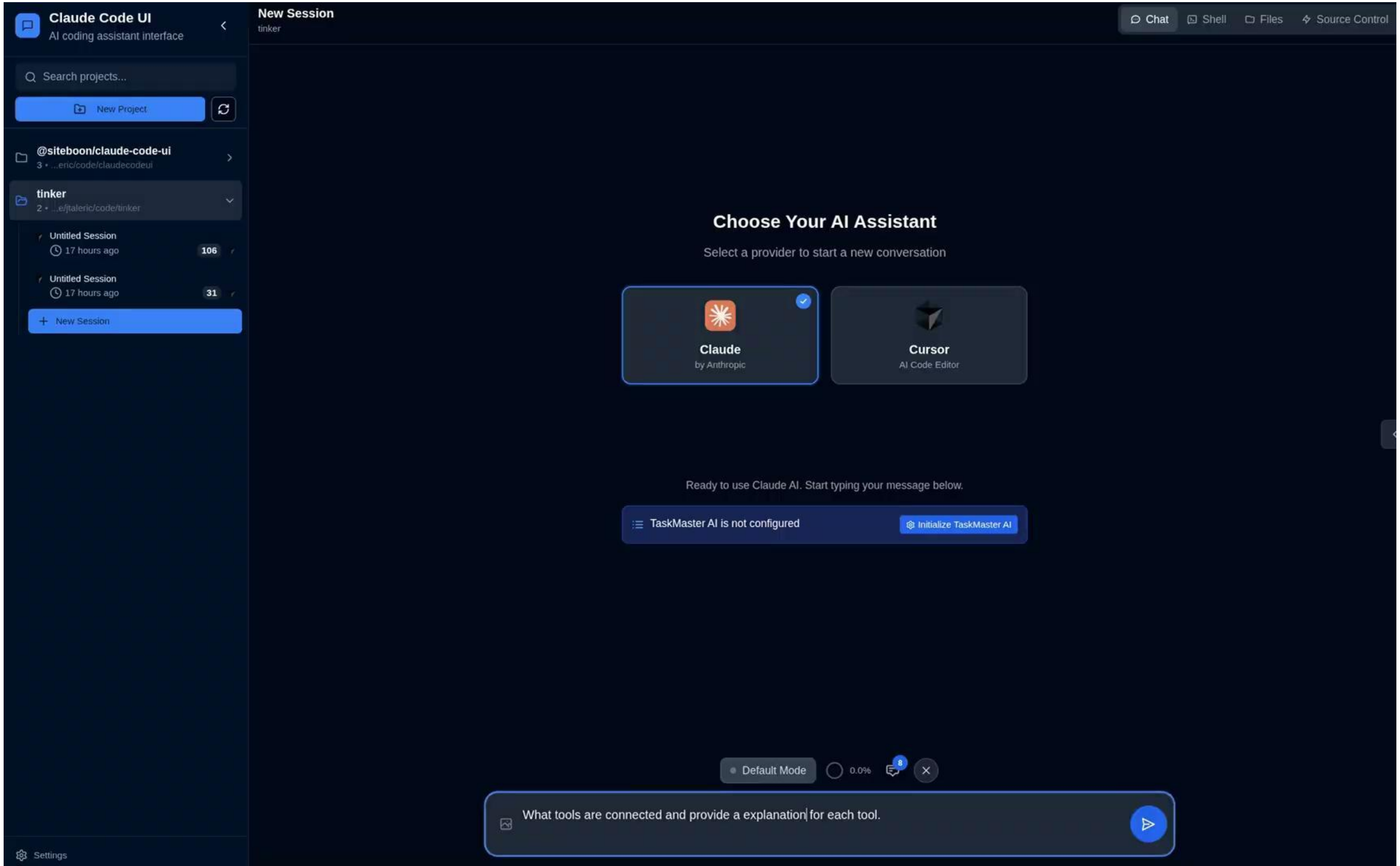
Now that we have established our workloads, and methodologies… The obvious question is, where is the AI?

We have recently introduced Orion-MCP, which allows developers and users to ask their favorite LLM that is connected to our MCP server, “Has OpenShift Regressed?”

This isn’t limited to simple requests, users can ask more complex questions, like “Has my PR introduced a regression” or for the LLM if OpenShift has regressed to look through the PR changelog and determine what PR could of possibly introduced the regression.


Query datalake for results that match fingerprint
Matching runs
Orion
JSON Result which is passed to the LLM
Orion MCP
Is UDN Performance regressing in OpenShift 4.21?
LLM
?
A

Orion-mcp with LLM - Demo



Choose Your AI Assistant

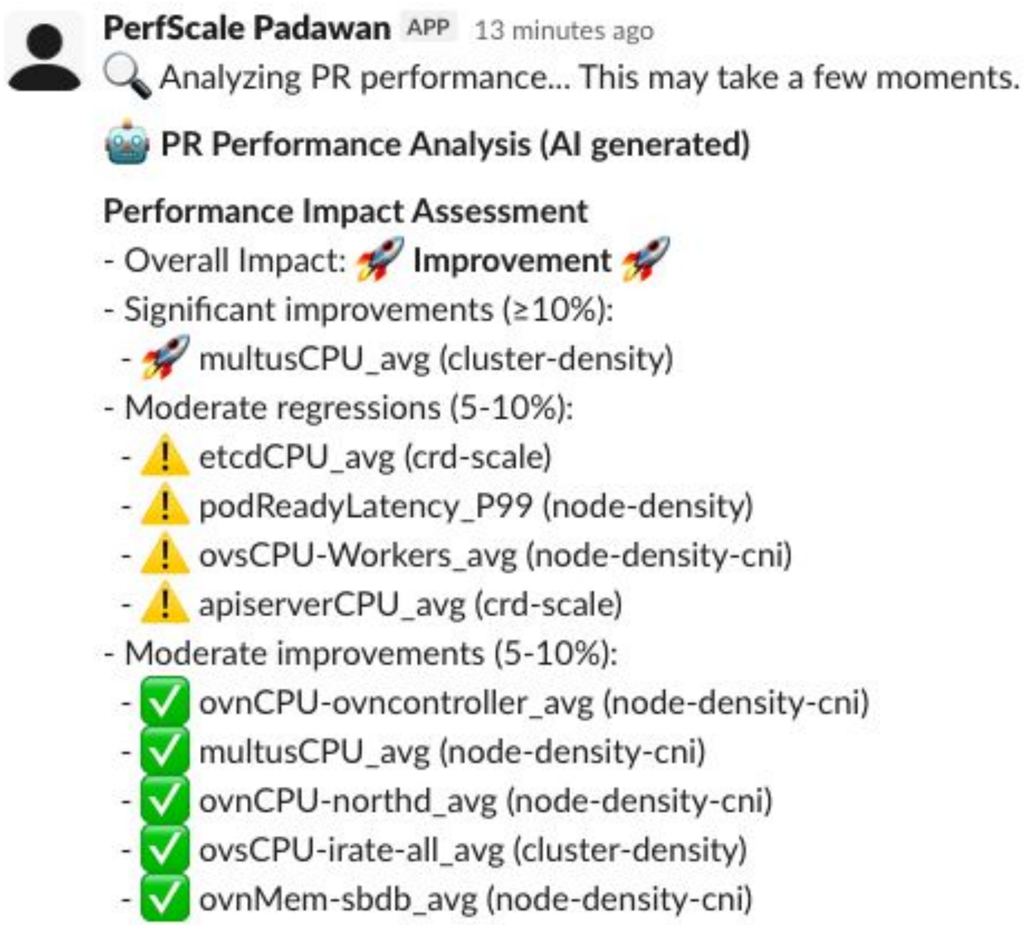
Select a provider to start a new conversation

Ready to use Claude AI. Start typing your message below.

TaskMaster AI is not configured

What tools are connected and provide a explanation for each tool.

Orion-mcp slack integration



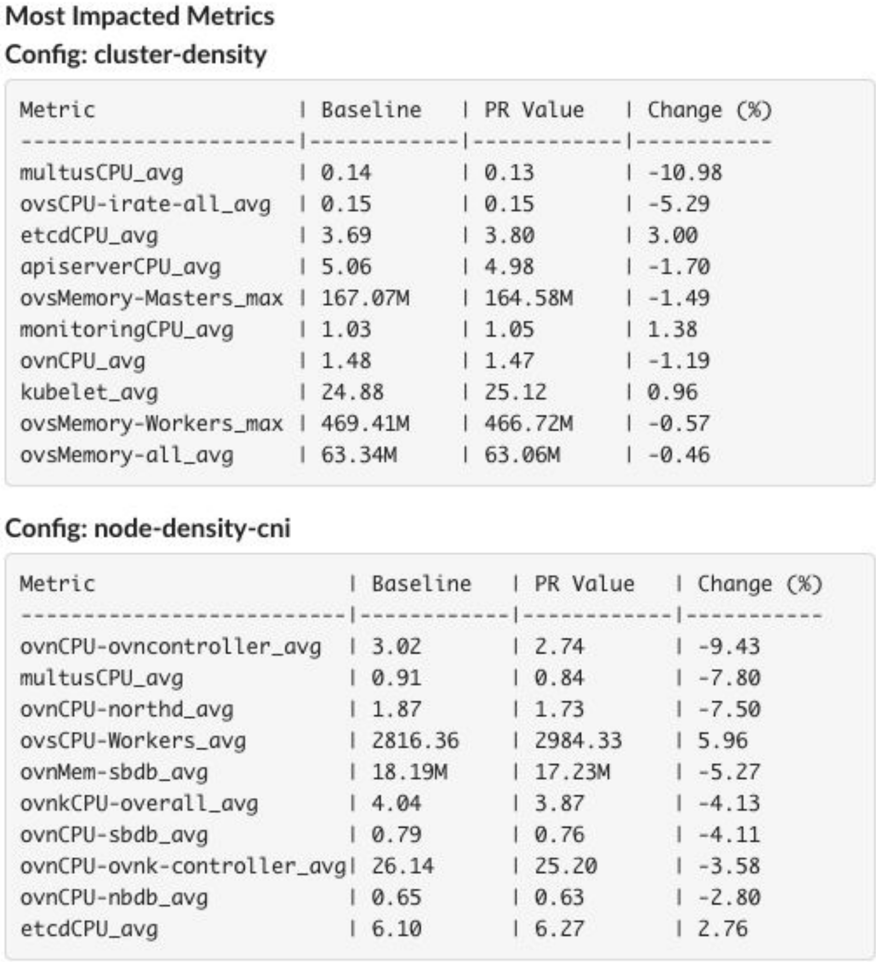
PerfScale Padawan APP 13 minutes ago

🔍 Analyzing PR performance... This may take a few moments.

🤖 **PR Performance Analysis (AI generated)**

Performance Impact Assessment

- Overall Impact: 🚀 **Improvement** 🚀
- Significant improvements (≥10%):
  - 🚀 multusCPU_avg (cluster-density)
- Moderate regressions (5-10%):
  - ⚠️ etcdCPU_avg (crd-scale)
  - ⚠️ podReadyLatency_P99 (node-density)
  - ⚠️ ovsCPU-Workers_avg (node-density-cni)
  - ⚠️ apiserverCPU_avg (crd-scale)
- Moderate improvements (5-10%):
  - ✅ ovnCPU-ovncontroller_avg (node-density-cni)
  - ✅ multusCPU_avg (node-density-cni)
  - ✅ ovnCPU-northd_avg (node-density-cni)
  - ✅ ovsCPU-irate-all_avg (cluster-density)
  - ✅ ovnMem-sbdb_avg (node-density-cni)



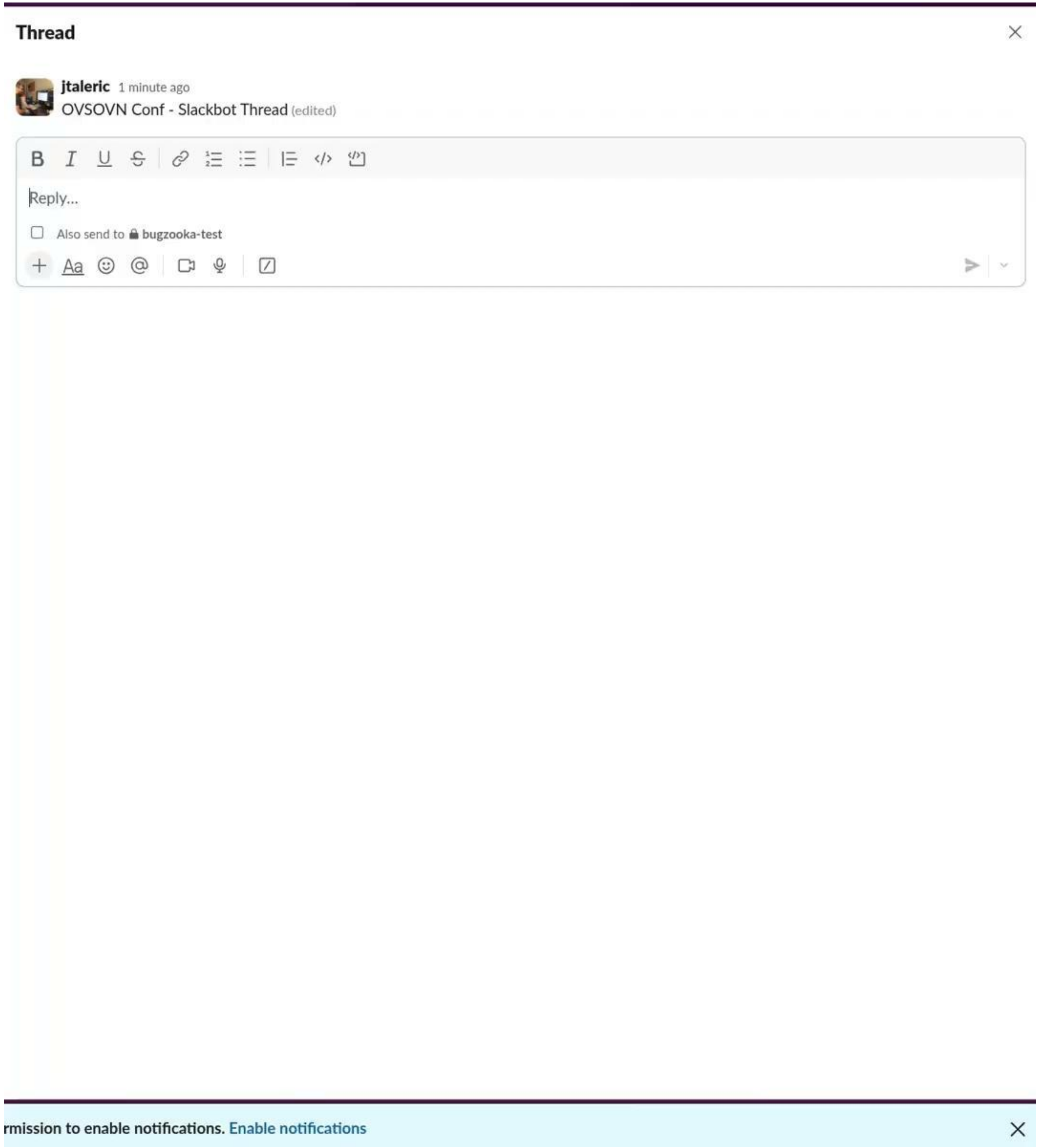
Most Impacted Metrics

Config: cluster-density

```
Metric                | Baseline    | PR Value    | Change (%)
---------------------|------------|------------|-----------
multusCPU_avg         | 0.14        | 0.13        | -10.98
ovsCPU-irate-all_avg  | 0.15        | 0.15        | -5.29
etcdCPU_avg           | 3.69        | 3.80        | 3.00
apiserverCPU_avg      | 5.06        | 4.98        | -1.70
ovsMemory-Masters_max | 167.07M     | 164.58M     | -1.49
monitoringCPU_avg     | 1.03        | 1.05        | 1.38
ovnCPU_avg            | 1.48        | 1.47        | -1.19
kubelet_avg           | 24.88       | 25.12       | 0.96
ovsMemory-Workers_max | 469.41M     | 466.72M     | -0.57
ovsMemory-all_avg     | 63.34M      | 63.06M      | -0.46
```

Config: node-density-cni

```
Metric                    | Baseline    | PR Value    | Change (%)
-------------------------|------------|------------|-----------
ovnCPU-ovncontroller_avg  | 3.02        | 2.74        | -9.43
multusCPU_avg             | 0.91        | 0.84        | -7.80
ovnCPU-northd_avg         | 1.87        | 1.73        | -7.50
ovsCPU-Workers_avg        | 2816.36     | 2984.33     | 5.96
ovnMem-sbdb_avg           | 18.19M      | 17.23M      | -5.27
ovnkCPU-overall_avg       | 4.04        | 3.87        | -4.13
ovnCPU-sbdb_avg           | 0.79        | 0.76        | -4.11
ovnCPU-ovnk-controller_avg| 26.14       | 25.20       | -3.58
ovnCPU-nbdb_avg           | 0.65        | 0.63        | -2.80
etcdCPU_avg               | 6.10        | 6.27        | 2.76
```



Thread

jtaleric 1 minute ago
OVSOVN Conf - Slackbot Thread (edited)

rmission to enable notifications. Enable notifications

Orion-mcp metadata (wip)

> Show me the kubeApi CPU usage of the last 5 results of OCP 4.19 on the daily virtualization bare metal runs and display the corresponding OVN, OVN-k8s and OVS versions,, without images.

Run	ovsCPU workers	**OVN**	**OVN-k8s**	**OVS**	OCP
1 (most recent)	2904	**25.09**	**1.1**	**3.11**	4.21
2	2910	**25.09**	**1.1**	**3.11**	4.21
3	2897	**25.03**	**1.0**	**3.11**	4.20
4	2910	**24.11**	**1.0**	**3.9**	4.19
5	2900	**24.11**	**1.0**	**3.9**	4.19

Challenges and next steps

- Consider adding to the ovn-ci tests one of our performance workloads to start identifying performance and scale issues upstream in ovn-kubernetes?
- Agentic AI workload for CPT
- Increase test coverage
  - I.e.: localnets, etc.
- More metadata context
- More integrations:
  - Slack
  - Github

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat