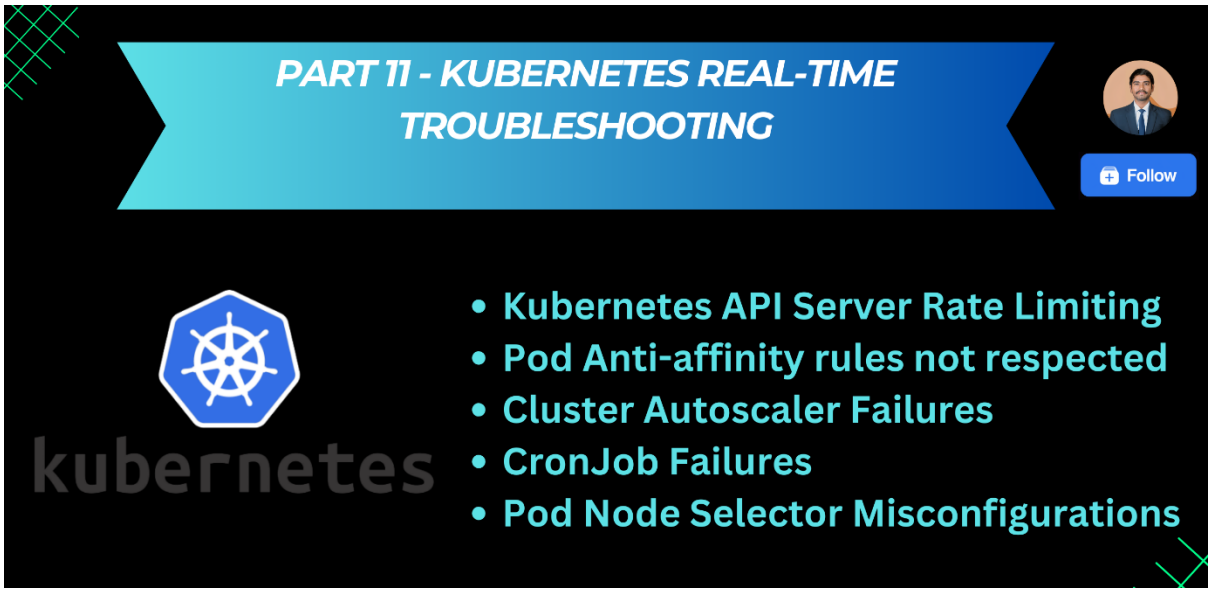
Part 11: Kubernetes Real-Time Troubleshooting

Introduction 🌐

Welcome to the world of Kubernetes troubleshooting, where every challenge is an opportunity to sharpen your skills and emerge victorious. Join us as we embark on a journey through common real-time scenarios, unraveling mysteries, and uncovering solutions along the way.



Scenario 51: Kubernetes API Server Rate Limiting

https://www.linkedin.com/in/prasad-suman-mohan

Symptoms: API server returns `429 Too Many Requests` errors due to rate limiting, causing client requests to fail or experience high latency.

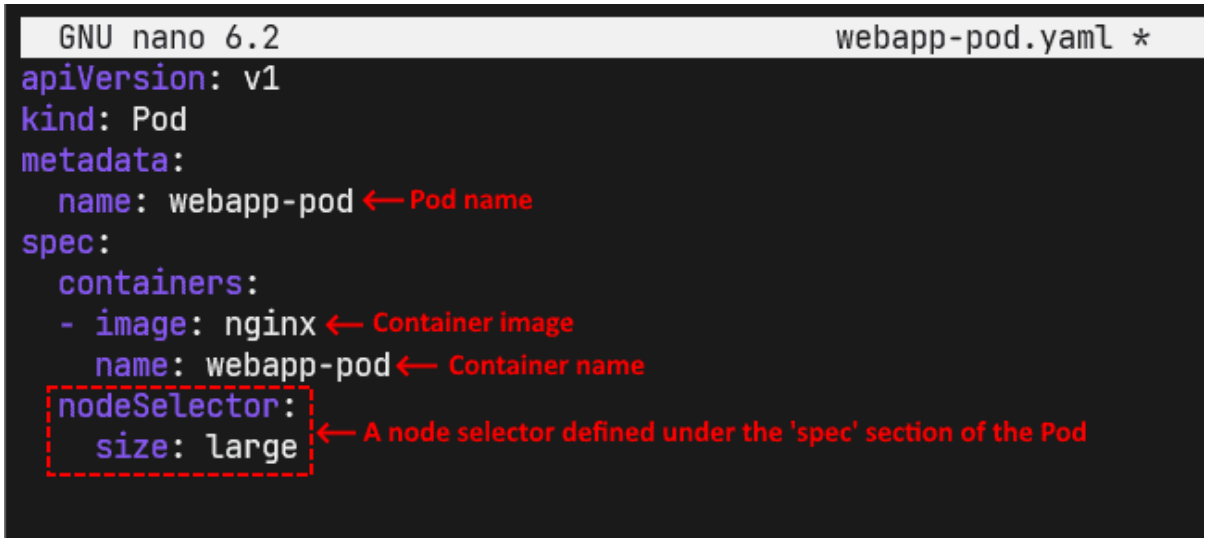
Diagnosis: Monitor API server metrics and logs (`kubectl get --raw /metrics`, `kubectl logs -n kube-system <api_server_pod>`) to identify rate limiting occurrences and affected clients.

```
},
response: {
  status: 429,
  statusText: 'Too Many Requests',
  headers: {
    date: 'Wed, 24 May 2023 16:15:36 GMT',
    'content-type': 'application/json',
    'content-length': '349',
    connection: 'keep-alive',
    'access-control-allow-origin': '*',
    'openai-model': 'gpt-3.5-turbo-0301',
    'openai-organization': 'niche-mates',
    'openai-processing-ms': '30008',
    'openai-version': '2020-10-01',
    'strict-transport-security': 'max-age=15724800; includeSubDomains',
    'x-ratelimit-limit-requests': '3500',
    'x-ratelimit-limit-tokens': '90000',
    'x-ratelimit-remaining-requests': '3498',
    'x-ratelimit-remaining-tokens': '89914',
    'x-ratelimit-reset-requests': '19ms',
    'x-ratelimit-reset-tokens': '56ms',
    'x-request-id': '0ddf4b34092d830cc9cff11bdaa8ce70',
    'cf-cache-status': 'DYNAMIC',
    server: 'cloudflare',
    'cf-ray': '7cc6d5df9ab90c3b-AMS',
    'alt-svc': 'h3=":443"; ma=86400, h3-29=":443"; ma=86400'
  },
```

Solution:

1. Optimize client applications and controllers to reduce the frequency and volume of API requests, implementing request batching or caching where possible.
2. Adjust API server rate limit configurations (`--max-requests-inflight`, `--max-mutating-requests-inflight`) to accommodate higher request loads.
3. Scale out the API server by adding more instances in a high-availability setup to distribute the request load and reduce rate limiting.
4. Use a dedicated client certificate with higher rate limits for critical or high-priority clients to ensure their requests are not throttled.

Scenario 51: Pod Anti-Affinity Rules Not Respected



Symptoms: Pods are scheduled on the same node despite having pod anti-affinity rules, leading to suboptimal distribution and potential single points of failure.

Diagnosis: Describe the affected pods (`kubectl describe pod <pod_name>`) and inspect the anti-affinity rules and node assignments to verify if the rules are being applied correctly.

Solution:

1. Review and adjust the pod anti-affinity rules in the pod templates to ensure they are correctly defined and feasible within the cluster's node resources.
2. Verify that the scheduler's configuration supports pod anti-affinity rules and is correctly prioritizing them during scheduling decisions.
3. Use taints and tolerations in conjunction with anti-affinity rules to achieve the desired pod distribution and placement across the cluster.
4. Scale out the cluster by adding more nodes to provide additional scheduling options and improve agreement to anti-affinity rules.

Scenario 53: Cluster Autoscaler Failures

Symptoms: Cluster autoscaler fails to add or remove nodes as needed, leading to resource shortages or excessive costs.

Diagnosis: Check the cluster autoscaler logs (`kubectl logs -n kube-system cluster-autoscaler`) and inspect node status and scaling events to identify failures or misconfigurations.

Solution:

1. Verify that the cluster autoscaler configuration matches the cloud provider's scaling policies and resource quotas.
2. Ensure that the autoscaler has the necessary permissions to create and delete nodes within the cluster.
3. Adjust the scaling parameters (e.g., min/max node counts, scale-down delay) to optimize the autoscaler's responsiveness and efficiency.
4. Monitor autoscaler performance and scaling events to identify and address any issues or bottlenecks in the scaling process.

Scenario 54: CronJob Failures

Symptoms: CronJobs fail to execute as scheduled, leading to missed tasks or inconsistent execution.

Diagnosis: Check the CronJob status and events (`kubectl get cronjob`, `kubectl describe cronjob <cronjob_name>`) and inspect the logs of the failed job pods (`kubectl logs <pod_name>`).

kubernetes/kubernetes

#108023 Cronjob fails to be scheduled - reports JobAlreadyActive even...

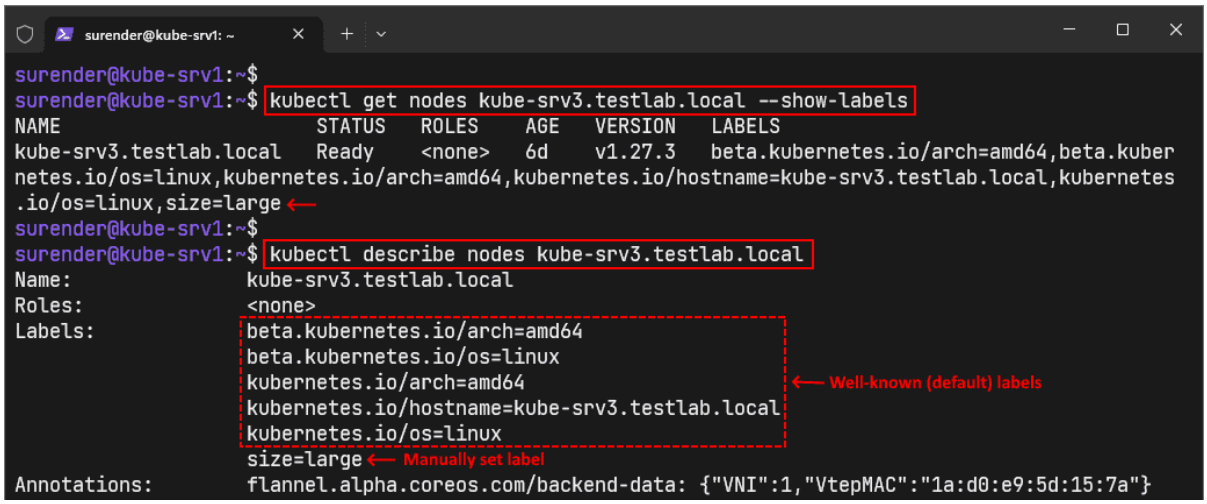
6 comments

simonjpartridge opened on February 9, 2022

Solution:

1. Ensure that the CronJob schedule syntax is correct and conforms to the expected cron format.
2. Verify that the job template specified in the CronJob is correctly defined and contains valid configurations.
3. Check for resource constraints or quotas that may prevent job pods from being scheduled or running successfully.
4. Review and adjust CronJob concurrency policies (`spec.concurrencyPolicy`) to manage overlapping executions and prevent conflicts.

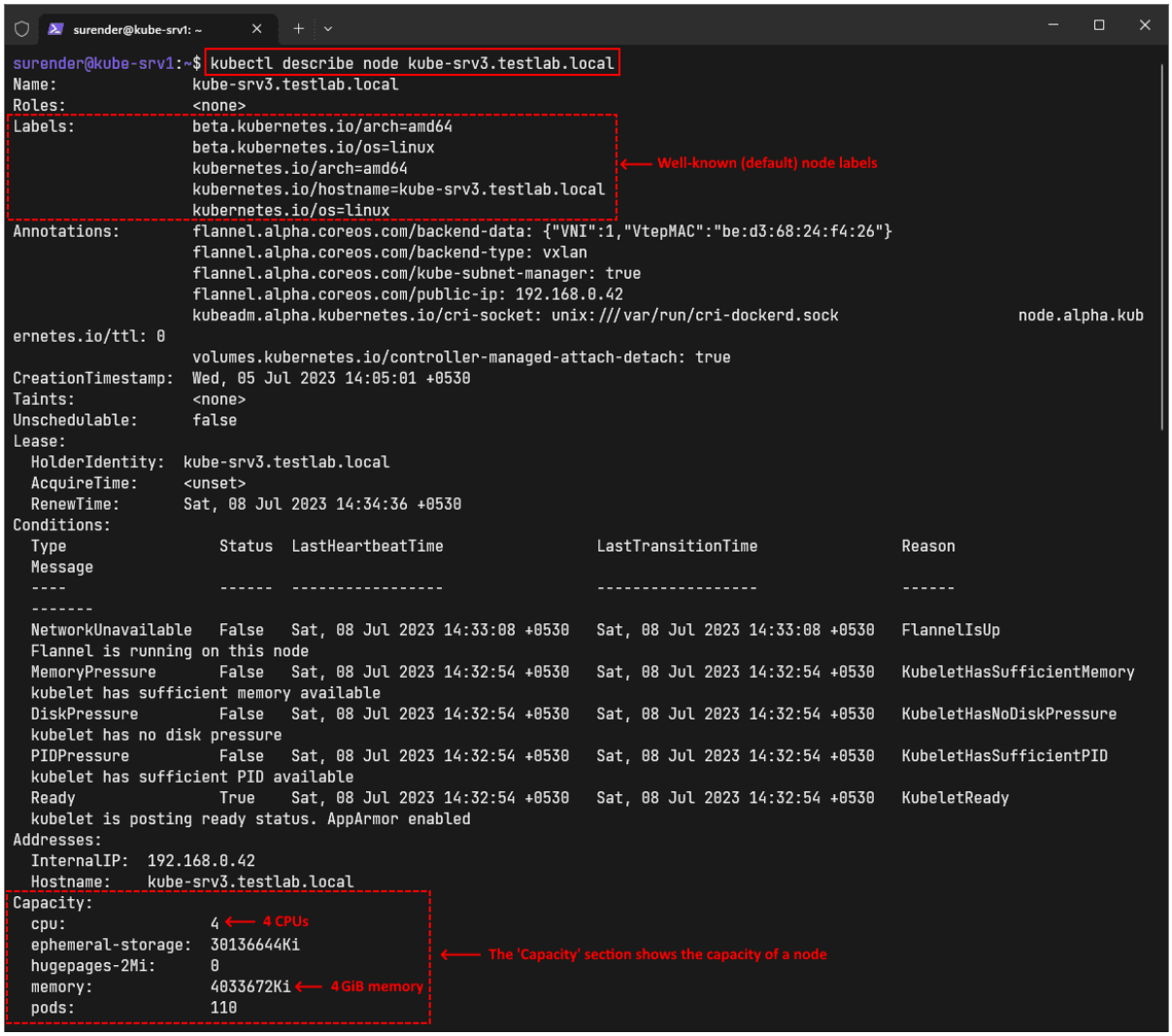
Scenario 55: Pod Node Selector Misconfiguration



Symptoms: Pods remain in a pending state due to unsatisfiable node selector requirements, causing deployment delays.

Diagnosis: Describe the affected pods (`kubectl describe pod <pod_name>`) and inspect the node selector and node labels for mismatches.

Solution:



1. Verify that the node selector in the pod specification matches the labels on the intended nodes.
2. Check for any node taints that may prevent pods from being scheduled and apply corresponding tolerations in the pod specification.
3. Adjust the node selector or labels on the nodes to align with the deployment requirements and facilitate pod scheduling.
4. Use node affinity rules to provide more flexible and advanced scheduling constraints based on node characteristics.

In the up-coming parts, we will discussion on more troubleshooting steps for the different Kubernetes based scenarios. So, stay tuned for the and follow @Prasad Suman Mohan for more such posts.

https://www.linkedin.com/in/prasad-suman-mohan