This runbook provides step-by-step guidance to investigate and mitigate a spike in Compute Service Processing Unit (CSPU) usage for a specific category node. A spike in CSPU can indicate resource exhaustion, increased load, inefficient processes, or underlying hardware contention.

Trigger:
This runbook should be followed when a monitoring alert indicates CSPU usage on a compute category node exceeds the defined threshold (e.g., 85% or higher for over 5 minutes).

Checklist Before Starting:
- Confirm access to OCI metrics and logs
- Ensure access to affected compute nodes via SSH
- Have escalation contacts for service owners ready

Step 1: Verify the Alert
- Review the CSPU spike alert details
- Confirm the affected availabilityDomain, instanceId, and category

```
Command : perl CopyEdit oci monitoring metric-data summarize-metrics-data \
--namespace oci_computeagent \ --query-text "CSPUUtilization[1m].max()" \
--resource-id <instance_id> \ --start-time $(date -u -d "-10 minutes"
+%Y-%m-%dT%H:%M:%SZ) \ --end-time $(date -u +%Y-%m-%dT%H:%M:%SZ)\
```

Step 2: Identify Noisy Neighbors or Process Load
- SSH into the affected instance (use bastion if needed)
- Run CPU and memory inspection

```
Command : kotlin CopyEdit ssh opc@<public_ip> top -o %CPU
```

Look for:
- Unexpected high CPU usage processes
- Runaway or zombie processes

```
Optional: perl CopyEdit ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head
```

Step 3: Check System and Kernel Logs
- Look for kernel or OS-level alerts such as throttling, OOM killer, or thermal issues

```
Command: bash CopyEdit dmesg | tail -n 50 journalctl -p 3 -xb
```

Step 4: Inspect Load Balancing and Autoscaling
- Check if this category node is disproportionately receiving traffic
- Review load balancer logs or autoscaling rules
- If autoscaling is enabled, verify if a scale-out event occurred

```
Command (OCI CLI): pgsql CopyEdit oci autoscaling configuration list oci
load-balancer lb list
```

Step 5: Investigate Recent Deployments or Job Triggers
- Check for recent application or job deployments that may be compute-heavy
- Correlate time of spike with deployment times

```
Command: perl CopyEdit git log --since="30 minutes ago" --pretty=format:"%h %an %s
Or
```

check Jenkins/CI/CD dashboard for recent jobs targeting that node pool or category.

Step 6: Apply Immediate Mitigation (If Required)

- If a specific process is confirmed as the root cause, consider killing or throttling it

Command: bash CopyEdit kill -9 <PID>
Or
temporarily isolate the instance (for non-critical nodes): css CopyEdit oci compute instance action --instance-id <instance_id> --action STOP
Only do this with change approval if service disruption is possible.

Step 7: Notify Relevant Teams

- Inform the owning service team of the impacted node category
- Share metrics, logs, and steps taken so far
- Use the designated incident Slack or Teams channel

Suggested summary: pgsql CopyEdit CSPU spike detected on node <instance_id> in AD1. Top process: java (PID 1322, 92% CPU). No scaling occurred. Load balanced traffic appears uneven. Service owner engaged.

Step 8: Monitor for Stability

- After mitigation or traffic redistribution, continue monitoring CSPU for the next 30 minutes
- Ensure usage remains under 80%
- Confirm that alerts auto-resolve or silence them manually if verified stable

Command: perl CopyEdit oci monitoring alarm-state update \ --alarm-id <alarm_ocid> \ --alarm-state OK

Step 9: Document the Incident

- Record the timeline of events
- Capture the root cause, impacted systems, and mitigation actions
- Submit post-incident report to your standard tracking tool (e.g., Jira, ServiceNow)