Root Cause Analysis:

Rally down due to users unable to login

The following is a detailed accounting of the service outage that Rally users experienced on June 11 and 12, 2019 at ~7:30am MST.

Root Cause Analysis Summary

Event Date	06/11/2019 and 06/12/2019
Event Start	Day 1 - 0726, Day 2 - 0733
Time Detected	Day 1 - 0726, Day 2 - 0733
Time Resolved	Day 1 - 0746, Day 2 - 0739
Event End Time	Day 1 - 0802, Day 2 - 0750
Root Cause	Both PaaS VIPs pointed to single Router and that Router became unavailable and the VIPs did not fail over
Customer Impact	Users could not login and use the system and those in the system would have limited functionality.

Issue:

On May June 11th and June 12th Rally experienced 2 related outages. The downtime for June 11th was 20 minutes and for June 12th it was 6 minutes. The outage was due to a failure of one of our internal load balancers that route traffic to our authentication service.

Currently internal traffic to our authentication system is routed through 2 virtual IP's pointing to the load balancer cluster. Normally these 2 IP's are routed to 2 different load balancer instances. During the outage both virtual IP's pointed to the same load balancer instance which subsequently failed and caused all authentication requests to timeout.

Due to a misconfiguration of keepalived when the load balancer instance failed traffic was not automatically routed to one of the healthy load balancer nodes. Manual intervention was required in order to restore the system to a healthy state.

The outage on June 12th was a repeat of the same issue. The Rally Operations team was in the process of investigating the first incident and did not have a complete understanding of the failure scenario in order to implement any remediations.

Cause:

After further investigation we have a better understanding of what caused both VIP's to point to the same load balancer node.

The default configuration of Openshift, our PaaS platform, by default does not prevent having multiple router nodes and VIP's from pointing to the same load balancer. About 2 years ago we broke out the VIP's into separate services in order to allow us to make configuration changes without downtime. This, combined with the default configuration, allowed for both VIP's to point to the same load balancer. When that load balancer failed all requests being serviced by the PaaS failed with it.

Preventative actions:

In order to prevent this from happening in the future we have made a couple of configuration changes. The first of these is to deploy a 4th router node. The next change that we have made is to configure our 4 router nodes into pairs with each VIP assigned to a pair. Additionally we have added pod affinity rules that will prevent the members of each router pair from pointing to the same load balancer. This configuration will allow the failure of a VIP, router node or load balancer to not affect requests being serviced by the PaaS.