Continuum High Availability Architecture and Strategies

Overview

This document serves as a guideline for utilizing and practicing a high-availability architecture that is designed to reduce downtime and mitigate risks of data loss. Specifics on how to implement each strategy may be provided in referenced URLs to other pages. If your organization needs specific guidance and assistance contact your VersionOne account representative for additional help from the VersionOne Services team.

Backup and Restore Process

The number one practice that most often gets overlooked in an HA architecture is designing a repeatable and automated backup process. There are two database components and a few optional file system directories that should be under consideration for backup.

First, MongoDB database contains most of the configuration and state data that the Continuum application needs to restore properly. The standard "database dump" processes should be sufficient to capture this data for storage and use if needed. See the following link for more information on scripting out a backup process.

https://community.versionone.com/VersionOne_Continuum/Continuum_Installation_and_Maintenance/Continuum_Maintenance_and_Configuration/Backing_Up_and_Restoring_Continuum

Directories on the Continuum server that should be considered adding to a backup process are as follows:

- /etc/continuum - configuration files with system settings and database connectivity settings. Backing up this directory will speed the restore process
- /var/continuum/canvas - if custom canvas files are used, these should be backed up
- /var/continuum/log - this directory is optional, the logs are for historical purposes only but can add considerable size to the backup

Any backup files produced as part of an automated process should be stored off the server and possibly in another data center or using a cloud object store. It is recommended that the backup process run at least once a day during a slow period, however downtime is not required to perform the backups.
Once the backup process is automated and scheduled then next thing to do is practice and document the restore process. It is one thing to have backups, it is another to know how to use the backups when they are needed and to make sure that more than just the primary application support personnel know how to use them. Documenting the backup and restore process and publishing this document is very important. Document the different restore processes in case of a single database loss all the way to a total server loss and replacement process.

The link to the documentation above has the commands for scripting a restore process for the databases and files from a tar archive. The actual commands can vary based on the brand of Linux as well as possibly the versions of the database servers. Practice on spare servers

---

**Monitor Disk Space**

The number one cause of Continuum downtime is lack of disk space. Consumption of disk space within Continuum is proportional to the number of repositories under management as well as the amount of developer commits and interfaces to third party tools. From an architecture standpoint, disk space should be considered as Continuum usage grows and added appropriately. I.T. infrastructure monitoring tool should be used to monitor disk space when it approaches a threshold and add space accordingly.

---

**Configure the Messenger**

Receiving notifications from Continuum will help head off small problems before they get bigger. A list of Administrator emails can be configured to receive these alerts.

For more information on configuring the Continuum Messenger:

https://community.versionone.com/Ver...Administration

---

**Use DNS**

All in-bound server requests to the Continuum application should use domain names that can be easily mapped to new server addresses. Requests (users, API calls, commit webhooks, etc.) should never use IP addresses or specific server names or addresses. This practice also extends to Continuum addresses within the application itself if the services reside on multiple machines (see next topic).

---

**Separation of (Server) Duties**

Long a preferred practice of traditional I.T. operations splitting Continuum over multiple servers has benefits, but also complicates the support process. A typical scenario would involve putting the MongoDB database on its own server. This allows for the server specifications and disk settings to be tailored for the specific database application use case.

The Continuum services themselves can be spread across multiple servers. In practice this is done less frequently, but in those cases where it makes sense that is discussed further in sections to follow. Just keep in mind as the architecture...
spreads out over multiple servers the backup and restore process gets more complicated and requires detailed documentation and practice.

One thing to be aware of when splitting the Continuum services and data stores onto different machines: upgrading Continuum to a new release will be different than the normal upgrade process. The upgrade may need to be performed on every machine that has Continuum software and will need to be planned accordingly.

---

**MongoDB Replication**

MongoDB has its own fault tolerance configuration called Replication.

[https://docs.mongodb.com/manual/replication](https://docs.mongodb.com/manual/replication)

---

**Load Balancing HTTP Servers**

The Continuum HTTP server, also called the ctm-ui service, can be duplicated across several nodes and served from behind either a hardware or software based HTTP load balancer. Sticky sessions should be enabled for user session persistence. Enabling this feature varies based on the load balancer, however conceptual information can be found at the link below.

[https://en.wikipedia.org/wiki/Load_balancing_(computing)#Persistence](https://en.wikipedia.org/wiki/Load_balancing_(computing)#Persistence)

Another valid reason for load balancing the Continuum HTTP server is for the acceptance of commit data from repositories via web hooks. For high volume development organizations spreading the processing of web hook data over several servers and having redundancy in case of an outage is critical to reducing the possibility of a delay in processing or outage.

---

**Multiple Automate Task Servers**

The Continuum Automate Task Engine server processes can be spread over two or more servers for redundancy and load balancing purposes. This may not be a factor until the Continuum environment is using a lot of Automate Tasks to perform automation. However once this is the case it makes sense to move Automate onto its own server and possibly install it on multiple servers. Automate has its own internal load balancer that parcels out Task Instance work to the Task Engine server with the lowest load percentage at that moment. When one Task Engine server fails to respond, it is taken out of the balancing algorithm.