Provisioning Server

WebRTC Virtual Classroom PLatform

Table of Contents

Overall design 2

Provisioning Server 2

Node control 2

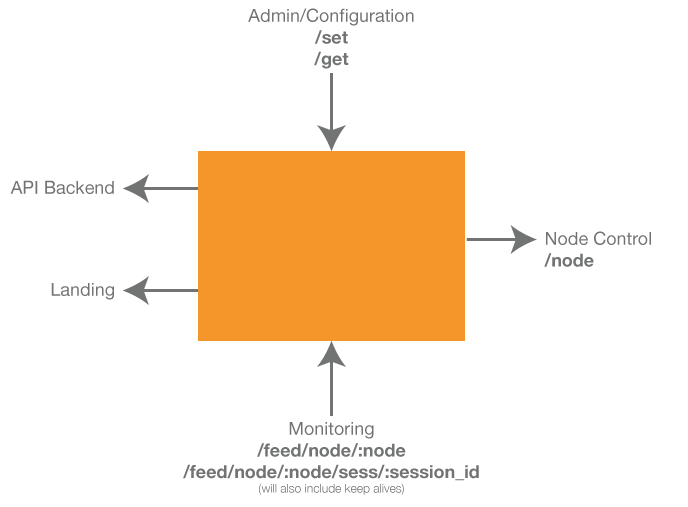
Admin command & configuration 3

Monitoring & Keepalives 3

# Overall design

Insert design diagram here.

# Provisioning Server



The provisioning server’s free body diagram above shows 5 interfaces. The rest of the API section for the provisioning server will conform to this view.

## Node control

The following operations need to be performed on the node:

|  |  |  |
| --- | --- | --- |
| Category | API | Comment |
| SESSION | Start a session | The full class configuration and run time information is provided in the request body |
| SESSION | Stop a session | Stop a running or about to start class. This would be called in the case of a manual intervention. Under normal circumstances, the session will manage it self and terminate itself. |
| NODE | Update a session | Update some information mid-way, say a log server going down and another one coming up (later) |
| NODE | Get status | Should return a big JSON object containing all it’s information (maybe including all sessions information as well) |
| NODE | Acquire | Acquire a node when told about a new node (via the admin command). A new node is brought up with no configuration effort. The provisioning is then told about the node (its IP Address). The provisioning server then invokes the ‘acquire’ API on the target node, supplying it with all the information it needs. |
| NODE | Modify | Update node specific information like change of log server etc. |

### Start A Session

## Admin command & configuration

The following operations need to be supported:

|  |  |  |
| --- | --- | --- |
| Category | API | Comment |
| LOGIN | Authorization | While the login could re-use the auth gateway, the authorization (in the beginning) can happen via a direct call to the DB, or via command line script. We don’t need an elaborate admin panel for user management. |
| CONFIG | Set resources | A catch all API to set all resources, including nodes, logs servers, persistent storage etc. |
| CONFIG | Set core configuration | Set’s the core configuration, including things like the address of the api-backend etc. |
| ADMIN | Stop session | Admin intervention to terminate a session |
| ADMIN | Get session information | Get information regarding the status of any particular (or all) sessions running on a specific (or all) node(s). |
| ADMIN | Get node information | Get node detailed information |
| CONFIG | Dump Config | Dump config into a JSON file |
| CONFIG | Restore Config | Restore config from a JSON file |

## Monitoring & Keepalives

The following operations need to be supported:

|  |  |  |
| --- | --- | --- |
| Category | API | Comment |
| PING | Ping | Ping the Provisioning server periodically. Should the provisioning server Pong back? |
| Feedback | Node health | Periodic node health status (still debating whether this should be a push or a pull. If the node goes down these messages will not come. Which means, the provisioning will have to ‘guess’ that Oh! No message since X minutes, so possibly the node is down. This implies that the provisioning will have to maintain a timer. So if it \_has\_ to, then why not the provisioning server itself poll for health? That way the logic of determining if a node is down become very natural). This applies to the ping/pong scenario as well. |
| Feedback | Session status | Including all state transitions |
| Feedback | Async Notifications | Event driven asynchronous notifications, usually for fatal errors. |

## Landing

The following operations need to be supported:

|  |  |  |
| --- | --- | --- |
| Category | API | Comment |
| SESS\_INFO | Get session info | Called on class joining. |

# Appendix A – Data maintained

The Provisioning server ought to maintain the following data sets, in fast as well as persistent storage:

|  |  |
| --- | --- |
| NODE Status | |
| Core Resource Caps | Configured CPU, Memory, I/F Bandwidth |
| Core Resources Usage | CPU, Memory, Network BW |
| List of sessions | List of session IDs |
| Configuration | IP Address, |
| Core services status | Status of core services like docker, fluent, etc |
| Overall Status | Up, Down or Unknown |
| Firmware version | Which version of the session are we running this node on (or should it be on a per session basis?) |

|  |  |
| --- | --- |
| RESOURCE Configuration | |
| Log Servers | Multiple log servers (fail safe redundancy) |
| AV configuration | * Media servers address(es) – mostly same for Tokbox * Secret keys, licensing information * Chrome/Firefox extension URL |
| Content | * Content Server address * Amazon keys etc |
| Chat | * Server address * Any keys etc |
| Session artifacts persistent storage | * Persistent Storage server (maybe content server?) * Any information to access/use it |
| Docker repository | * URL for the repository * Credentials |