Web UI requirement

*We will require all source files HTML/CSS/JS as part of the deliverable*

# Main Menu on the left pane

Always visible in all pages

1. Dashboard
2. Create/Update
3. Queue
4. Summary
5. Upgrades
6. About

# Dashboard page

Collects utilization data across multiple clouds and visualizes them in various graphics

## Global drop down to choose which cloud is being queried

Entries are “All Clouds” or individual cloud names returned by query

API:

/api/v1/clouds/summary

Sample Response:

{

[

“name”: “Development”,

“description”: “Cloud used during coding, unit test phase”

],

[

“name”: “Testing”,

“description”: “Cloud used for sanities, functional, and interaction testing”

],

[

“name”: “Staging”,

“description”: “Staging cloud”

],

[

“name”: “Production”,

“description”: “Production cloud”

]

}

## Context creation timeline graph

depicts on Y-axis number of Contexts created at an instance of time (on X-axis). Time range should be adjustable through mouse scroll.

API:

/api/v1/contexts/timeRangeTrend?startTime=2023-08-24T22:30:00.000Z&endTime=2023-08-24T23:30:00.000Z

Sample response:

{

[

“startTime”: “2023-08-24T22:30:00.000Z”,

“endTime”: “2023-08-24T22:45:00.000Z”,

“numContexts”: 10

],

[

“startTime”: “2023-08-24T22:45:00.000Z”,

“endTime”: “2023-08-24T23:00:00.000Z”,

“numContexts”: 2

],

[

“startTime”: “2023-08-24T23:00:00.000Z”,

“endTime”: “2023-08-24T23:15:00.000Z”,

“numContexts”: 4

],

[

“startTime”: “2023-08-24T23:15:00.000Z”,

“endTime”: “2023-08-24T23:30:00.000Z”,

“numContexts”: 0

]

}

## Context creation Pie Chart

Tile showing Pie chart of Contexts created. Chart should have a toggle button “Normalize to Max”. User is allowed to create max number of entities. Pie chart shows Contexts “created successfully”, “failed”, “in progress”. If “Normalize to Max” is selected Pie chart 100% corresponds to “Max Allowed Contexts”. If not selected Pie chart 100% corresponds to “Total number of contexts created”.

API:

/api/v1/contexts/summary

Sample Response (without max):

{

[

“success”: 5,

“inProgress”: 2,

“failed”: 1

“services”: {

“success”: 6,

“inProgress”: 1,

“failed”: 1

},

“deployments”: {

“success”: 5,

“inProgress”: 2,

“failed”: 1

},

“pods”: {

“success”: 14,

“inProgress”: 4,

“failed”: 2

}

]

}

Here, 100% is 8 and 5 (success) is shown in green, 2 (inProgress) in amber, 1 (failed) in red.

Sample Response (with max):

{

[

“maxAllowed”: 50,

“success”: 5,

“inProgress”: 2,

“failed”: 1

“services”: {

“success”: 6,

“inProgress”: 1,

“failed”: 1

},

“deployments”: {

“success”: 5,

“inProgress”: 2,

“failed”: 1

},

“pods”: {

“success”: 14,

“inProgress”: 4,

“failed”: 2

}

]

}

Here, 100% is 50 and 5 (success) is shown in green, 2 (inProgress) in amber, 1 (failed) in red. Rest [50 – (5+1+2) =] 42 (remaining contexts that can be created) is shown in grey.

## Legend with kubernetes resource stats

Next to Pie chart in the same tile show following legend:

Services: Green Box #success; Amber Box #inProgress; Red Box #failed

Deployments: Green Box #success; Amber Box #inProgress; Red Box #failed

Pods: Green Box #success; Amber Box #inProgress; Red Box #failed

Timeline graphs of rps(requests per second), latency (microseconds), cpu (vcpu), and memory (MB).

API:

/api/v1/stats/summary

Sample response

{

[

“startTime”: “2023-08-24T22:30:00.000Z”,

“endTime”: “2023-08-24T22:45:00.000Z”,

“requestsPerSecond”: 20,

“latency”: 10,

“cpu”: 0.002,

“mem”: 30

],

[

“startTime”: “2023-08-24T22:45:00.000Z”,

“endTime”: “2023-08-24T23:00:00.000Z”,

“requestsPerSecond”: 50,

“latency”: 40,

“cpu”: 0.005,

“mem”: 60

],

[

“startTime”: “2023-08-24T23:00:00.000Z”,

“endTime”: “2023-08-24T23:15:00.000Z”,

“requestsPerSecond”: 10,

“latency”: 5,

“cpu”: 0.001,

“mem”: 20

],

[

“startTime”: “2023-08-24T23:15:00.000Z”,

“endTime”: “2023-08-24T23:30:00.000Z”,

“requestsPerSecond”: 100,

“latency”: 70,

“cpu”: 0.01,

“mem”: 80

]

}

# Create Context Page

Main frame is split in to two panes with Right Pan only having Load and Save buttons. Left pane having the form to create a context

## Load/Save (submit buttons)

Two buttons On the right of the form pane to “Load” “Save”

“Load” looks up if context with name exists and loads its parameters

Method Get /api/v1/contexts?name=contextName

“Save” collects the current data provided by user and does POST request to create context

Method POST /api/v1/contexts/create

with Json payload with all the info provided in the form as described below

## Context Name

Input text box (has validation of allowed input characters; alphanumeric and – [hyphen])

## Feature dropdown

UI queries backend for available features and populates dropdown.

Method GET /api/v1/features/list

“Choose Feature …” feature dropdown lists features as options in alphabetical order

Selection of feature sends a query to backend to get all services that support the feature

Method GET /api/v1/features/services?featureName=feat1

Sample Response:

{

[

“serviceName”: “svc1”

],

[

“serviceName”: “svc2”

],

[

“serviceName”: “svc3”

],

[

“serviceName”: “svc4”

]

}

## Service Specification section

Given the number of services in a feature, that many such line entries of the following type need to be created to take input from the user.

<label serviceName></label>: <input text>

## Add Rules section

Pane to add and provide parameters for rules for context creation

### Rule type dropdown

Dropdown default “Choose Rule Type …”; Only HTTP Route option is allowed currently

If HTTP Route is selected, show

### HTTP Route Match

Dropdown with match types as options:

1. Header
2. Path
3. Query Params

*If “Header” or “Query Params” is selected:* Show 2 input textboxes and 1 dropdown

1. Name
2. Value
3. Type (Dropdown: Exact or Regex)

*If “Path” is selected:* Show 1 input textboxes and 1 dropdown

1. Value
2. Type (Dropdown: Exact or Regex)

# Context Queue / Results

Table with the following fields:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No. | Context Name | Owner (userid) | Cloud | Status |
| 1 | Ctx 1 | User1 | Development | Running |
| 2 | Ctx 2 | User2 | Testing | Queued |
| 3 | Ctx 3 | User1 | Production | Completed |

# Context Deployment in Clouds

## Left pane with Cloud and contexts:

Text area having Cloud Name followed by Contexts deployed in it; Each cloud is listed one below the other in the text area.

## Right Pane with feature/context details table

Shows status features deployed, context that deploys it and status.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Context** | **PR** | **Status** |
| Feat #1 | Ctx 1 | Link to PR 1123 | Success |
| Feat #2 | Ctx 2 | Link to PR 1342 | Failed |
| Feat #3 | Ctx 3 | Link to PR 1654 | Success |
| Feat #4 | Ctx 4 | Link to PR 987 | In Progress |

API:

/api/v1/features/contexts

Sample response:

{

[

“featureName”: “Feat #1”,

“description”: “Feature number 1”,

“contextName”: “Ctx 1”,

“Status”: “Success”,

“pullRequest”: <LINK\_TO\_PR>[1123]

],

[

“featureName”: “Feat #2”,

“description”: “Feature number 2”,

“contextName”: “Ctx 2”,

“Status”: “Failed”,

“pullRequest”: <LINK\_TO\_PR>[1342]

],

[

“featureName”: “Feat #3”,

“description”: “Feature number 3”,

“contextName”: “Ctx 3”,

“Status”: “Success”,

“pullRequest”: <LINK\_TO\_PR>[1654]

],

[

“featureName”: “Feat #4”,

“description”: “Feature number 4”,

“contextName”: “Ctx 4”,

“Status”: “In Progress”,

“pullRequest”: <LINK\_TO\_PR>[987]

]

}

# Upgrades

Page to look for and upgrade to specific version of software. Upgrade button is grayed initially.

## Validate connectivity to server

Software provider URL with credentials to validate if eligible for upgrades

## Check for updates

Check for updates. Returns updates that current user is eligible for.

## Version

Dropdown of available versions. On choosing version, provide link release notes. Upgrade button is enabled.

# About

Provide current software version of framework being used by various cloud deployments.