# Budget API

The Budget API allows you to create, update, and delete recurring budget rules to control your Kubernetes spending. Weekly and monthly budgets can be established on namespaces, clusters, and labels to set limits on cost spend, with the option to configure alerts for reaching specified budget thresholds via email, Slack, or Microsoft Teams.

## Budget API

`POST` `http://<your-kubecost-address>/model/budget`

Creates a recurring budget rule or updates a recurring budget rule when provided the ID of the existing rule.

### Request Body

| Name | Type | Description |
|---|---|---|
| name* | string | Name of the budget rule |
| values* | string | Used for specifying the group and name where the budget rule is applied to in the form of a key-value pair. Accepts `namespace`, `cluster`, or `label` for the first value, followed by the corresponding item. For example, when applying a budget rule to a namespace named `kubecost`, this parameter is configured as `values=namespace:kubecost`. |
| interval* | string | The interval that the budget will reset with (either `weekly` or `monthly`). |
| intervalDay | int | The day the budget will reset. When `interval=weekly`, `intervalDay` is the day of the week, with `intervalDay=0` for Sunday, `intervalDay=1` for Monday, etc. When `interval=monthly`, `intervalDay` corresponds with the day of the month. |

| Name | Type | Description |
|---|---|---|
| spendLimit* | int | The budget limit value. |
| id | string | Only should be used when updating a budget rule; ID of the budget rule being modified. For more info, see the Using the `id` parameter section below. |
| action | string | Optional configurations for providing visibility when your budget exceeds a specified percentage threshold. This parameter can generate emails, and Slack or Microsoft Teams messages to suit your work environment. For more information, see the Using Budget Actions section below. |

200: OK

```
{
    "code": 200,
    "data": [
        {
            "name": "<budget name>",
            "id": "<budget id>",
            "values": {
                "<namespace or cluster>": [
                    "<name of namespace of cluster>"
                ]
            },
            "kind": "",
            "interval": "",
            "intervalDay": ,
            "spendLimit": ,
            "actions": [
                {
                    "amount": 0,
                    "percentage": 1,
                    "slackWebhooks": [],
                    "msTeamsWebhooks": [],
                    "emails": [],
                    "lastFired": ""
                }
            ],
            "window": {
                "start": "",
                "end": ""
            },
            "currentSpend":
        }
}
```

## Get recurring budget rule(s)

`GET` `http://<your-kubecost-address>/model/budgets`

Lists all existing recurring budget rules

200: OK

```
{
    "code": 200,
    "data": [
        {
            "name": "<budget name>",
            "id": "<budget id>",
            "values": {
                "<namespace or cluster>": [
                    "<name of namespace of cluster>"
                ]
            },
            "kind": "",
            "interval": "",
            "intervalDay": ,
            "spendLimit": ,
            "actions": [
                {
                    "percentage": 1,
                    "slackWebhooks": [],
                    "msTeamsWebhooks": [],
                    "emails": [],
                    "lastFired": ""
                }
            ],
            "window": {
                "start": "",
                "end": ""
            },
            "currentSpend":
        }
}
```

## Delete recurring budget rule

`DELETE` `https://<your-kubecost-address>/model/deleteBudget`

Deletes a budget rule defined by `id`

#### Path Parameters

| Name | Type | Description |
|---|---|---|
| id* | string | ID of the recurring budget rule to be deleted |

200: OK

```
{
    "code": 200,
    "data": []
}
```

## Formatting parameters when creating/updating budget rules

Creating and updating recurring budget rules uses POST requests, which will require submitting a JSON object in the body of your request instead of adding parameters directly into the path (such as when deleting a recurring budget rule). See the Examples section below for more information on formatting your requests.

## Using the `id` parameter

The `id` parameter when using the endpoint `/budget` is considered optional, but its use will vary depending on whether you want to create or update a budget rule.

When creating a new budget rule, `id` should not be used. An ID for the budget rule will then be randomly generated in the response. When updating an existing budget rule, `id` needs to be used to identify which budget rule you want to modify, even if you only have one existing rule.

The `id` value of your recurring budget is needed to update or delete it. If you don't have the `id` value saved, you can retrieve it using `/budgets`, which will generate all existing budgets and their respective `id` values.

## Using Budget Actions

You can configure greater visibility towards tracking your budgets using the `actions` parameter, which will allow you to create alerts for when your budget spend has passed a specified percentage threshold, and send those alerts to you or your team via email, Slack, or Microsoft Teams.

When providing values for `actions`, `percentage` refers to the percentage of `spendLimit` which will result in an alert. For example, if `"spendLimit": 2000` is configured for a weekly budget rule and `"percentage": 50` is configured, an alert will be sent to all listed emails/webhooks if spending surpasses $1000 USD for the week.

```
"actions" : [
            {
                "percentage": 100,
                "slackWebhooks": [
                    "<example Slack webhook>"
                ],
                "emails": [
                    "foo@kubecost.com",
                    "bar@kubecost.com"
                ],
                "msTeamsWebhooks": [
                    "<example Teams webhook>"
                ]
            }
        ]
```

## Configuring currency

Kubecost supports configuration of the following currency types: USD, AUD, BRL, CAD, CHF, CNY, DKK, EUR, GBP, IDR, INR, JPY, NOK, PLN, and SEK. Kubecost does *not* perform any currency conversion when switching currency types; it is for display purposes, therefore you should ideally match your currency type to the type in your original cloud bill(s).

Currency type can only be changed via a `helm` upgrade to your *values.yaml*, using the flag `.Values.kubecostProductConfigs.currencyCode`. For example, if you needed to convert your currency type to EUR, you would modify the helm flag as:

```
kubecostProductConfigs:
    currencyCode: EUR
```

## Examples

**Create a recurring budget rule for my test cluster which resets every Wednesday with a budget of $100.00 USD, and will send an alert via email when spending has exceeded 75% of the spend limit.**

```
curl --location '<your-kubecost-address>/model/budget' \
--header 'Content-Type: application/json' \
--data-raw '{
        "name": "budget-rule",
        "values": {
            "cluster":["test"]
        },
        "kind": "soft",
        "interval": "weekly",
        "intervalDay": 3,
        "spendLimit": 100,
        "actions" : [
            {
                "percentage": 75,
                "emails": [
                    "foo@kubecost.com",
                ]
            }
        ]
}'
```

**Create a recurring budget rule for my `kubecost` namespace which resets on the 1st of every month with a budget of $400.00 USD, and will send an alert via Slack and Microsoft Teams when spending has exceeded $100.00 of the spend limit.**

```
curl --location '<your-kubecost-address>/model/budget' \
--header 'Content-Type: application/json' \
--data-raw '{
        "name": "budget-rule-2",
        "values": {
            "namespace":["kubecost"]
        },
        "kind": "soft",
        "interval": "monthly",
        "intervalDay": 1,
        "spendLimit": 400,
        "actions" : [
             {
                "percentage": 25,
                "slackWebhooks": [
                    "<example Slack webhook>"
                ],
                "msTeamsWebhooks": [
                    "<example Teams webhook>"
                ]
            }
        ]
}'
```

## Use cases

For an example use case on how to use budgets to achieve proactive cost control, see here.