User Guide

Nobl9 User Guide

Back to Nobl9 Documentation

Welcome, and thank you for choosing Nobl9. The User Guide shows you how to use the Nobl9 platform. Learn to access your account, connect to data sources, develop alert policies, and configure the command-line tool. To better understand how everything works together view the use cases for examples.

Prerequisites

The following software, tools, or actions are needed to ensure a great onboarding experience:

1. Use a Mac, Linux, or Windows machine to run `sloctl`. The Nobl9 command-line utility makes it easier to create and update many SLOs at once.

2. Select a Kubernetes cluster or any Docker environment (or use a Docker environment on your local machine to start) to run the Nobl9 agent, which collects service level indicator metrics from your existing metrics system (such as Datadog, New Relic, or Prometheus).

3. Verify that you received an email from Nobl9 to set up a user account.

Set Up a User Account

As a Nobl9 customer, you will receive an invitation email with an activation link.

> *If you were invited to Nobl9 and did not receive an invitation email, please contact support@nobl9.com.*

1. Locate the Nobl9 user invitation sent to your email.

2. Click the link to accept the invitation and follow the instructions to set up your user account. After setting up your account, a confirmation page appears which asks you to return to the login screen.

3. Return to the login screen by visiting https://app.nobl9.com in your browser.

Logging into Nobl9 User Interface

You will need to log into the Nobl9 web user interface (UI) using the credentials created during the account setup.

1. Go to https://app.nobl9.com

2. Enter your email address and password created during the account setup or click Login with Google if you have a single sign-on (SSO) account.

Setting Up Nobl9 Command Line (sloctl)

The command-line interface (CLI) for Nobl9 is named `sloctl`. As a best practice, use the `sloctl` CLI when creating or updating multiple SLOs at once, creating or updating multiple thresholds, or when updating SLOs as part of CI/CD. The web user interface is available to give you an easy way to create and update SLOs, and to familiarize you with the features availale in Nobl9, while `sloctl` aims to provide a systematic and/or automated approach to maintaining SLOs as code.

Use the following steps to download the Nobl9 CLI.

1. Download the appropriate binary executable zip file from https://github.com/nobl9/sloctl/releases.

2. Extract the binary. The following are platform-specific examples:
   - **Mac**: Install Homebrew
     `brew tap nobl9/sloctl`
     `brew install sloctl`
     > 💡 ***Note:*** *The Mac operating system occasionally requires file permissions. When this happens, give the file executable permission. For further details, see Mac instructions.*

   - **Linux**: `/usr/local/bin`
     > 💡 ***Note:*** *The Linux operating system occasionally requires file permissions. When this happens, give the file executable permission. For further details, see Linux instructions.*

   - **Windows**: `C:\Windows\System32`
     > 💡 ***Note:*** *The Windows operating system occasionally requires file permissions. When this happens, give the file executable permission. For further details on copying to system folders and executable permissions, see Windows instructions.*

3. Copy and paste the following in a file in your home directory (recommended) or in the directory where you intend to run `sloctl`.
```
project: default
url: https://app.nobl9.com/api
oktaOrgURL: https://accounts.nobl9.com
oktaAuthServer: auseg9kiegWKEtJZC416
organization: YOUR_ORGANIZATION
```

4. Edit the file.

5. Create a **Client ID** and **Client Secret** pair for use in `sloctl`.
   - Navigate to **Settings** → **Access Keys** in the web UI.
   - Click **Create Access Key**.
   - Click the **Download credentials file**. You will need these credentials when you run the configure command in sloctl.

6. Configure `sloctl` to use the provided credentials.
   - Run `sloctl configure`.
   - Paste the **Client ID** and **Client Secret** from the web UI.

7. Test the configuration by entering `sloctl get slos` into the command line.
   > 💡 *If there are no SLOs created in your account, you might see this message:* `No resources found in default project.`
   > *The message means the configuration is correct. The command line will return a* `401 error` *if the configuration does not work.*

Data Sources

Use the **Direct** connection if you want Nobl9 to access your server by connecting directly over the internet. This method may be less secure as you will need to open the port the data source is running on for Nobl9 to connect.

Use the **Agent** method if you want to run an agent alongside your server. You will not need to directly expose your server to Nobl9, the agent will periodically connect to Nobl9 using an outbound connection.

Adding Data Sources for the Service

You can have multiple data sources for your service. Configure Nobl9 to connect to these (one or multiple) data sources to collect all service data in real-time.

Running data collection through an agent means that special inbound access to your network is not needed and Nobl9 doesn't have to store credentials to your other metric systems.

Use the following steps to define a data source and run a Nobl9 Agent.

1. Go to **Settings** icon in the Web UI and click to create a new integration.
2. Follow the on-screen instructions to run the agent.
   Recommendations:

   a. Samples are provided for a Kubernetes Deployment and a simple Docker run command.

   b. Run the agent(s) in production clusters or in a location that can access production metrics.

   c. Consider running the agent in your local Docker environment at first for ease of troubleshooting.

Adding Data Sources (Agent)

Use the **Agent** method if you want to run an agent alongside your server. You will not need to directly expose your server to Nobl9, the agent will periodically connect to Nobl9 using an outbound connection.

Amazon CloudWatch

1. Enter a **Project**.
   Use the Project when multiple users are spread across multiple teams or projects. When the Project field is left blank, a default value appears.
2. Enter a **Display name** (optional).
   The Display name allows you to enter a name with spaces.
3. Enter a **Name**.
   The Name is mandatory and can only contain lowercase, alphanumeric characters, and dashes. For example: `my-project-name`.
4. Enter a **Description**.
   Add the team, owner details, or the purpose of creating this specific integration.

AppDynamics

1. Enter the **Controller URL** to connect your data source (required). AppDynamics does not support user or agent requests that originate from any URL other than the **Controller URL**.
2. Enter a **Project** name.
   The **Project** field is intended for use in situation where multiple users are spread across multiple teams or projects. When **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-appdynamics-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Big Query

1. Enter a **Project** name.
   The **Project** field is intended for use in situation where multiple users are spread across multiple teams or projects. When **Project** field is left blank the typical **default** value appears.

2. The **Display Name** appears automatically when a name is entered into the **Name** field.

3. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-big-query-data-source`

4. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Datadog

1. Add the Datadog **API endpoint** to connect to your data source (required).

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-datadog-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Dynatrace

1. Add the **URL** to connect to your data source. Depending on the type of environment, the **URL** must match one of the following formats:

- SaaS: https://{your-environment-id}.live.dynatrace.com
- Managed: https://{your-domain}/e/{your-environment-id}
- Environment ActiveGate:https://{your-activegate-domain}/e/{your-environment-id}

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-dynatrace-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Elasticsearch

1. Add the **URL** to connect to your data source (required).

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-elasticsearch-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Grafana Loki

1. Enter the **URL**.
   The URL must start with http:// or https://.

2. Enter a **Project**.
   Use the Project when multiple users are spread across multiple teams or projects. When the Project field is left blank, a default value appears.

3. Enter a **Display name** (optional).
   The Display name allows you to enter a name with spaces.

4. Enter a **Name**.
   The Name is is mandatory and can only contain lowercase, alphanumeric characters, and dashes. For

example: my-project-name.

5. Enter a **Description**.
   Add the team, owner details, or the purpose of creating this specific integration.

Graphite

1. Add the **Graphite Render API URL** to connect to your data source (required). The URL must start with `http://` or `https://`.

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-graphite-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Lightstep

1. Enter the name **Lightstep organization** to connect to your data source (required).

2. Enter a name in the **Lightstep project** field.\

3. The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

4. The **Display Name** appears automatically when a name is entered into the **Name** field.

5. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-lightstep-data-source`

6. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

New Relic

1. Add the New Relic **Account ID** to connect to your data source (required).

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-new-relic-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Prometheus

1. Add the **URL** to connect to your data source (required).

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-prometheus-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Splunk

1. Enter the **URL** to connect to your data source (required).

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-splunk-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

> 💡 ***Note*** *For details on constructing Splunk queries go here.*

Splunk Observability

1. Add the **API Endpoint URL** to connect to your data source (required).

   For example: https://api..signalfx.com

2. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-splunk-observability-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

ThousandEyes

1. Enter a **Project** name.
   The **Project** field is intended for use in situation where multiple users are spread across multiple teams or projects. When **Project** field is left blank the typical **default** value appears.

2. The **Display Name** appears automatically when a name is entered into the **Name** field.

3. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-thousandeyes-data-source`

4. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Adding Data Sources (Direct)

Use the **Direct** connection if you want Nobl9 to access your server by connection directly over the internet. This method may be less secure, as you will need to open the port the data source is running on for Nobl9 to connect.

Amazon CloudWatch

1. Add **Access Key ID** and **Secret Access Key**.
   The Access Keys are user security credentials that are used to make programmatic calls to Amazon Web Services. These are the same credentials that you use to log into the AWS console. Both Access Key ID and Secret Access Key are created as a pair. Your Access Key ID and Secret Access Key are encrypted before being stored on the Nobl9 server.

> 💡 *Note: For more details, see here.*

1. Enter a **Project**.
   Use the Project when multiple users are spread across multiple teams or projects. When the Project field is left blank, a default value appears.

2. Enter a **Display name** (optional).
   The Display name allows you to enter a name with spaces.

3. Enter a **Name**.
   The Name is mandatory and can only contain lowercase, alphanumeric characters, and dashes. For example: `my-project-name`.

4. Enter a **Description**.
   Add the team, owner details, or the purpose of creating this specific integration.

AppDynamics

1. Enter the **Controller URL** to connect your data source (required). AppDynamics does not support user or agent requests that originate from any URL other than the **Controller URL**.

2. Enter your AppDynamics **ClientID**.

3. Enter your AppDynamics **Client Secret**.

4. Enter a **Project** name.
   The **Project** field is intended for use in situation where multiple users are spread across multiple teams or projects. When **Project** field is left blank the typical **default** value appears.

5. The **Display Name** appears automatically when a name is entered into the **Name** field.

6. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-direct-appdynamics-data-source`

7. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Big Query

1. Upload **Service Account Key File** to authenticate with Google Cloud.
   The file must be in JSON format. See the following reference.

2. Enter a **Project** name.
   The **Project** field is intended for use in situation where multiple users are spread across multiple teams or projects. When **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-direct-big-query-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Datadog

1. Add the Datadog **API endpoint** to connect to your data source (required).

2. Enter your Datadog **API Key**.
   For more information about how to manage your **API Key**, see *API and Application Keys*.

3. Enter your Datadog **Application Key**.
   For more information about how to manage your **Application Key**, see *API and Application Keys*.

4. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

5. The **Display Name** appears automatically when a name is entered into the **Name** field.

6. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-direct-datadog-data-source`

7. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

New Relic

1. Enter your New Relic **Account ID** to connect to your data source (required).

2. Enter the **Insights Query Key**.

3. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank, the typical **default** value appears.

4. The **Display Name** appears automatically when a name is entered into the **Name** field.

5. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-direct-new-relic-data-source`

6. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Splunk Observability

1. Add the **API Endpoint URL** to connect to your data source (required).

For example: https://api.REALM.signalfx.com

2. Enter the **Access Token** environment variable for authentication with the organization API Access Token.
   See Create and manage organization access tokens.

3. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

4. The **Display Name** appears automatically when a name is entered into the **Name** field.

5. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-direct-splunk-observability-data-source`

6. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

ThousandEyes

1. Enter the **OAuth Bearer Token** for authentication with the ThousandEyes API.
   When the Nobl9 agent is deployed for ThousandEyes, you must to provide a `THOUSANDEYES_OAUTH_BEARER_TOKEN` environment variable for authentication with ThousandEyes API.

   To get the **OAUTH_BEARER_TOKEN**:

   - Log in to your ThousandEyes account.
   - Navigate to **Account Settings**.
   - Select **Users and Roles**.
   - Navigate to the bottom of the page and you will see **User API Tokens**.
   - Select **OAuth Bearer Token**.
     Currently, Nobl9 only supports OAUTH_BEARER_TOKEN.

2. Enter a **Project** name.
   The **Project** field is intended for use in situation where multiple users are spread across multiple teams or projects. When **Project** field is left blank the typical **default** value appears.

3. The **Display Name** appears automatically when a name is entered into the **Name** field.

4. Enter a **Name** (required).
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `my-direct-thousandeyes-data-source`

5. Enter a **Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific data source. Adding a description can provide immediate context for any team member.

Services

Nobl9 uses services to represent distinct boundaries in your application. A service can be a user journey, internal or external API, or some other boundary—essentially anything you care about setting a service level objective for. For example, in a service desk application one service might be creating a new ticket. That service may rely on a user service, a queue, a notification service, and a database service, all of which could additionally be defined as additional services in Nobl9.

A service may be composed of other services. When adding a service, you can use labels to add additional metadata such as team ownership or upstream/downstream dependencies. Services can either be manually added in through the user interface or YAML, or automatically discovered from a data source based on rules.

A service can have one or more SLOs defined for it. Every SLO created in Nobl9 must be tied to a service.

Adding a Service

Perform the following steps in the in the **Manual Service** wizard to add a service. From the Web UI:

1. Navigate to the **Services** icon and click the + icon to start the **Manual Service** wizard.

2. Enter or create a project in the **Project Field**.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

3. Enter a **Display Name**.
   The Display Name is optional. You can use spaces. Nobl9 translates the Display Name field into a Kubernetes-style naming convention.

4. Enter a **Name**.
   A name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: `my-project-name`

5. Enter a **Description** with the team or owner details and the purpose of creating this specific service.

   Adding a description can provide immediate context for any team member.

Alert Methods

When an incident is triggered, Nobl9 enables you to send the alert to a notification engine or tool (for example, PagerDuty). Nobl9 also supports integration with a web endpoint by using webhooks where you define the endpoint and parameters to pass.

Adding Discord

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the ⊕ icon to add Discord.

3. Enter a **URL**.

4. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

5. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

6. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-discord-notification

7. Enter a **Description**.
   As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

Adding Email

Nobl9 can send alert notifications via email. You can add the recipients of the alert (To, CC, and BCC), customize the Subject and the body of the email.

1. Enter a **Project**. This field is intended for use in situations where multiple users are spread across multiple teams or Projects. When the **Project** field is left blank, a **default** value appears.

2. Enter a **Display name** (optional). You can use spaces. Nobl9 translates the **Display name** field into a Kubernetes-style convention.

3. The **Name** is required. It can only contain lowercase, alphanumeric characters, and dashes. For example, `my-project-name`.

4. The **Description** field is optional. You can add the team and owner details or explain the purpose of creating this alert. Adding a description can provide immediate context for your team members.

Compose email alert

1. Enter the recipients of the email alert. You can add an email address as a direct recipient (To), as a carbon copy (CC), or as a blind carbon copy (BCC).
   - You must provide at least one recipient in any one of these categories.
   - The maximum number of recipients for each category (To, CC, BCC) is 10. You can paste a list of recipients here. Email addresses must be separated with white space, comma (`,`) or semicolon (`;`). For example:
     - `email1@example.com email2@example.com email3@example.com email4@example.com`
     - `email1@example.com,email2@example.com,email3@example.com, email4@example.com`
     - `email1@example.com;email2@example.com;email3@example.com;email4@example.com`
     - `email1@example.com ; email2@example.com,email3@example.com email4@example.com`

2. By default, the **Subject** shows `Your SLO $slo_name needs attention!`.
   - You can add Noble9 variables (e.g., Project Name, Service Name, SLO Name, Alert Policy Name, Severity, etc.) to your alert **Subject**.
   - To add variables, click the ➕ button.

   > 💡 ***Note**: The character limit in this field is 90.*

3. The **Message** field is the body of your email alert. Nobl9 provides the following template of the body that you customize using plain text:

```
$alert_policy_name has triggered with the following conditions:
$alert_policy_conditions[]

Time: $timestamp
Severity: $severity
Project: $project_name
Service: $service_name
Organization: $organization
```

- Just as in the **Subject** field, you can add Noble9 variables (e.g., Project Name, Service Name, SLO Name, Alert Policy Name, Severity, etc.) to your alert **Message**.
- To add variables to your **Message** body, click the ➕ button.

> 💡 ***Note**: The character limit in this field is 2000.*

Adding Jira

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the ➕ icon to add Jira.

3. Enter a **URL**.

4. Enter a **Username**.
   The username is the email owner of the API Token.

   For example: jira-alerts@mycompany.com

5. Enter an **API Token**. The **API Token** is created by logging into your Jira account and clicking on your profile. The **API Token** is secret and is not returned with the `GET` command in sloctl. During creation, the **API Token** field is required and becomes optional when the alert is updated. The updated alert uses the behavior from the existing object.

6. Enter a **Jira Project Key** name.
   Jira Project Key is the code of the project.

   For example: AT (alert test), PM (project management)

7. The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

8. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

9. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-jira-notification

10. Enter a **Description**.
    As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

- The user must have access and permissions to create an issue in a project. The Nobl9 backend receives an error from Jira API, and the ticket will not be created.

- Creating the ticket fails when the project settings require some mandatory fields. For example, each ticket must have a *Due date* field value or must have the field set to *Fix versions*.
- The Nobl9 Jira message contains values for the following fields:
  - Summary
  - Description
  - IssueType (set always to **Bug**)
  - When creating a ticket, if any other field is required (other than **Summary**, **Description**, or **IssueType**) the ticket will fail.

Adding MS Teams

1. The **URL** must start with https:// for the MSTeams configuration.
2. Use the **Project** when multiple users are spread across multiple teams or projects. When the Project field is left blank, a **default** value appears.
3. The **Display name** is optional. You can use spaces. Nobl9 translates the **Display name** field into a Kubernetes-style convention.
4. The **Name** is required and can only contain lowercase, alphanumeric characters, and dashes. For example, `my-project-name`.
5. As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific integration.

Adding Opsgenie

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the ⊕ icon to add Jira.

3. Enter a **URL**.

4. Choose **Basic** or **GenieKey** from the **User Authentication Method** drop-down list.
   The **Basic** requires you to paste the `base64` hash created from the API Key.

   The default is **GenieKey** and requires the API Key to be obtained directly from the Opsgenie panel. (This method is easier use. The choice between the two authentication methods is offered for convenience, as some users may only have access to one of the methods.)

5. Enter an **API Key**. You will need to paste the **API Key** from Opsgenie panel.

6. The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

7. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

8. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-opsgenie-notification-key

9. Enter a **Description**.
   As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

Adding PagerDuty

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the **+** icon to add PagerDuty.

3. Enter a **Integration Key** for the PagerDuty configuration.
   PagerDuty requires a 32-character **Integration Key**.

4. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

5. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

6. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-pagerduty-notification

7. Enter a **Description**.
   As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

Adding ServiceNow

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the **+** icon to add ServiceNow.

3. Enter your ServiceNow **Username**.

4. Enter your ServiceNow **Password**.

5. Enter an **Instance ID**.
   The **Instance ID** is a globally unique ID across all ServiceNow instances. Check the `/stats.do` page to view the **InstanceID** for any instance.

6. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

7. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

8. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-servicenow-notification.

9. Enter a **Description**.
   As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

Adding Slack

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the ⊕ icon to add Slack.

3. Enter a **URL** for the Slack configuration. For Slack, it is an *Incoming Webhook URL*. It must start with `https://`.
    - For details on where to find your *Incoming Webhook URL* go here: Incoming webhooks for Slack.
    - If you need further help, contact your Slack administrator.

4. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

5. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

6. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-slack-notification.

7. Enter a **Description**.
   As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

Adding Webhook

1. Navigate to the **Settings** icon in the left-navigation menu. In the **Integrations** screen, select the **Alert Methods** tab.

2. Click the ⊕ icon to add a Webhook.

3. Enter a **URL** for the Webhook configuration. The **URL** must start with `https://`.

4. Enter a **Project** name.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

5. Enter a **Display name** (optional).
   The **Display name** allows you to enter a name with spaces.

6. Enter a the **Name** for your configuration.
   The name is required and can only contain lowercase, alphanumeric characters, and dashes.

   For example: my-webhook-notification.

7. Enter a **Description**.
   As a best practice, in the **Description** add the team, owner details, or the purpose of creating this specific configuration.

   > 💡 *Customers who use the Nobl9 webhooks may need to use an allow list for IP addresses from where the webhook calls are made. The IP addresses are as follows:*
   >
   > - *18.159.114.21*
   > - *18.158.132.186*

   - *3.64.154.26*

8. Add a **Custom Header** by clicking the **+ ADD** icon. Add your own customized headers as part of REST API call. Custom headers are included with the standard, auto-generated headers created by Nobl9.

9. Choose to add a **custom** or standard **Nobl9** notification in the **Notification Details** tab.

   - A **custom** Webhook lets you create your own payload in JSON format. (This functionality is only valid with JSON.)
     - Most fields are in string format, and the field must be inserted between double quotes (“”).
     - When a field has a suffix (array) on the list, you must use the formatting for a JSON array. The array fields have a special format with square brackets after the variable name.

       Example: $alert_policy_conditions[]

   - A standard **Nobl9** Webhook configuration lets you choose fields from a checklist, which is then sent with a default Nobl9 Webhook message. Select at least one field from the checklist.
     - project_name
     - service_name
     - organization
     - alert_policy_name
     - alert_policy_description
     - alert_policy_conditions[]
     - alert_policy_conditions_text
     - severity
     - slo_name
     - slo_details_link
     - experience_name
     - timestam

Service Level Objectives

A service level objective (SLO) is a target value or range of values for a service level that is measured by a service level indicator (SLI).

SLOs are just data. The most important thing to keep in mind if you adopted the principles of an SLO-based approach is that the ultimate goal is to provide you with a new dataset based on which you can have better discussions and make better decisions. There are no hard-and-fast rules: using SLOs is an approach that helps you think about your service from a new perspective, not a strict ideology. Nothing is ever perfect, and that includes the data SLOs provide you; it’s better than raw telemetry, but don’t expect it to be flawless. SLOs guide you, they don’t demand anything from you.[1]

[1] Hidalgo, *Implementing Service Level Objectives*, page 36.

Creating an SLO

In the Web UI, follow the five-step configuration in the **SLO Wizard** to create an SLO.

1. Navigate to **Service Level Objectives** icon and click the icon to start the **SLO Wizard**.

2. Select a **Service** from the drop-down list to tag the service for the SLO.

3. Click the **Data Source** drop-down list in the **Select Data Source and Metric** tab and choose a data source.

4. Select a type of **Metric**. and enter a **Query**.
   - A **Threshold Metric** is a single time series evaluated against a threshold.
   - A **Ratio Metric** allows you to enter two time series to compare (for example, a count of good requests and total requests).
5. Enter a **Query**, **Good Query**, or **Total Query** for the metric you selected.

   The following are query examples:

   - Threshold metric query for Datadog:
     ```
     avg:trace.http.request.duration{service:my-service}
     .as_count()
     ```
   - Ratio metric query for Datadog:

     Good Query:
     ```
     avg:trace.http.request.hits
     .by_http_status{service:my-service,!http.status_class:5xx}
     .as_count()
     ```
     Total Query:
     ```
     avg:trace.http.request.hits
     .by_http_status{service:my-service}.as_count()
     ```

   The following are examples of what can be queried. For more information, see the Use Case section in the User Guide.

Description	Result
Web service or API	HTTPS responses with 2xx and 3xx status codes.
In a queue consumer	Successful processing of a message.
In a serverless and function-based architectures	successful completion of an invocation.
In batch	normal exit (for example, `rc == 0`) of the driving process or script.
In a browser application	completion of a user action without JavaScript errors.

1. Choose a **Rolling** or **Calendar-Aligned** time window in the **Define Time Window** section.
   - Rolling time windows are better for tracking *recent* user experience of a service.
   - Calendar-aligned windows are best suited for SLOs that are intended to map to business metrics that are measured on calendar-aligned basis, such as every calendar month, or every quarter.
2. **Define Error Budget Calculation and Objectives**. Click the drop-down list in **Error Budget Calculation Method** and select either **Occurrences** or **Time Slices**.

Occurrences count good attempts against the count of all attempts. Since total attempts are fewer during low-traffic periods, it automatically adjusts to lower traffic volume.

- Use the **Target** to define the percentage of good events to total events that meet the defined target for the SLI.

Time Slices measure how many good minutes were achieved (minutes where a system is operating within defined boundaries), compared to the total minutes in the window. The disadvantage is that a bad minute that occurs during a low-traffic period (say, in the middle of the night for most of our users, when they are unlikely to even notice a performance issue) would penalize the SLO the same amount as a bad minute during peak traffic times.

- Use the **Target** to define the percentage of good events to total events that meet the defined target for the SLI.
- **Example**: A threshold that requires 95% of requests to complete within 100ms. Name the experience "Laggy".

```
    budgetTarget: 0.95
    displayName: Laggy
    value: 100
```

3. Name your objective in the **Add Name, Alert Policy & Tags** section.

   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.

   For example: `streaming-latency-slo`

4. Click the drop-down list next to **Alert Policies** to set an alert. If no alerts were created, navigate to the **Alert Policies** page and click the + icon to start the **Alert Policy Wizard**.

5. Enter a **Description**. Document relevant details or metadata for the SLI and SLO as description. As a best practice, it is recommended to add the team or owner details and the purpose of creating this specific SLO. This can provide an immediate context about the SLO to any team member.

Alerts

An alert policy expresses a set of conditions you want to track or monitor. The conditions for an alert policy define what is monitored and when to activate an alert. Set up an alert policy in using **the ALERT POLICY WIZARD** in the Web UI.

1. Select the **Alerts** icon and click the + icon to start the **ALERTS POLICY WIZARD.**

2. **Define an Alert Condition** by selecting one or more of the boxes.

   A **Defined Alert Condition** monitors the behavior and volatility of a data source. You can set a maximum of three alert conditions. Create another alert policy if you want to set more than three alert conditions.

   The **Error budget** relies on the targets set up in your SLA and SLOs. Error budgets measure the maximum amount of time a system can fail without repercussions.

   The **Remaining error budget** is the amount leftover from the error budget set up in the SLO.

The **Error budget burn rate** measures how fast the error budget is disappearing. The numbers in the error budget burn rate must match the numbers in the error budget.

3. Go to the **Add Alert Policy Name and Severity** tab.

   a. Enter a **Project**.
   The **Project** field is intended for use in situations where multiple users are spread across multiple teams or projects. When the **Project** field is left blank the typical **default** value appears.

   b. Enter a **Display name** (optional).
   The **Display Name** field allows you to enter a name with spaces.

   c. Enter a **Name** for the alert.
   A **Name** is required because metadata names are unique within each project and are validated against some RFC and DNS names. The name must contain only lowercase alphanumeric characters and dashes.
   For example: streaming-latency-alert

   d. Set the **Severity** to high, medium, or low.
   - **High**: A critical incident with a very high impact.
   - **Medium**: A major incident with significant impact.
   - **Low**: A minor incident with low impact.

   e. Create an **Alert Policy Description** (optional).
   Add the team or owner details and explain the purpose of creating this specific alert. Adding a description can provide immediate context for any team member.

4. Go to Select Alert Method and select the box to set up a webhook.

   Set up the integration in YAML using `sloctl` to apply changes. The webhook integration will then be available in the Alert Wizard in the Web UI.

Use Cases of SLO Configurations

The following examples explain how to create SLOs for sample services using sloctl.

A Typical Example of a Latency SLO for a RESTful Service

First, we want to pick an appropriate service level indicator to measure the latency of response from a RESTful service. In this example, let's assume our service runs in NGINX web server, and we're going to use a threshold-based approach to define acceptable behavior. For example, we want the service to respond in a certain amount of time.

> 💡 ***Note:*** *There are many ways to measure application performance. In this case we're giving an example of server-side measurement at the application layer (NGINX) but it might be advantageous for your application to measure this differently. For example, you might choose to measure performance at the client, or at the load balancer, or somewhere else. Your choice depends on what you are trying to measure or improve, as well as what data is currently available as usable metrics for the SLI.*

The threshold approach uses a single query, and we set thresholds or breaking points on the results from that query to define the boundaries of acceptable behavior. In the SLO YAML, we specify the indicator like this:

```
  indicator:
     metricSource:
       name: my-prometheus-instance
```

```
        project: default
      rawMetric:
        prometheus:
          promql: server_requestMsec{job="nginx"}
```

In this example use Prometheus. The concepts are similar for other metrics stores. We recommend running the query against your Prometheus instance and reviewing the result data, so you can verify that the query returns what you expect, and so that you understand the units: whether it's returning latencies as milliseconds or fractions of a second, for example. This query seems to return data between 60 and 150 milliseconds with some occasional outliers.

Choosing a Time Window

We need to choose whether we want a rolling or calendar-aligned window.

- Calendar-aligned windows are best suited for SLOs that are intended to map to business metrics that are measured on calendar-aligned basis, such as every calendar month, or every quarter.
- Rolling windows are better for tracking "recent" user experience of a service.

For our RESTful service, we'll be using the Rolling window SLO primarily to measure recent user experience and to make decisions about the risk of changes, releases, and how best to invest our engineering resources on a week-to-week or sprint-to-sprint basis. We want the "recent" period that we're measuring to trail back long enough that our users would consider it recent behavior. We choose to go with a 28-day window, which has the advantage of containing an equal number of weekend days and weekdays as it rolls.

```
  timeWindows:
    - count: 28
      isRolling: true
      period:
        begin: "2020-12-01T00:00:00Z"
      unit: Day
```

Choosing a Budgeting Method

There are two budgeting methods to choose from: **Time Slices** and **Occurrences**.

Time Slices
In the Time Slices method, what we count (objective we measure) is how many good minutes were achieved (minutes where our system is operating within defined boundaries), compared to the total minutes in the window.

This is useful for some scenarios, but it has a disadvantage when we're looking at "recent user experience" as we are with this SLO. The disadvantage is that a bad minute that occurs during a low-traffic period (say, in the middle of the night for most of our users, when they are unlikely to even notice a performance issue) would penalize the SLO the same amount as a bad minute during peak traffic times.

Occurrences
The Occurrences method is well suited to this situation. Occurrences count good attempts (in this example, requests that are within defined boundaries) against the count of all attempts (this means all requests, including requests that perform outside of defined boundaries). Since total attempts are fewer during low-traffic periods, it automatically adjusts to lower traffic volume.

```
    budgetingMethod: Occurrences
```

Establishing Thresholds

In this example we've talked to our product and support teams and can establish the following thresholds:

- The service has to respond fast enough that users don't see any lag in the web applications that use this service
- Our Product Manager thinks that 100ms (1/10th of a second) is a reasonable threshold for what qualifies at Okay latency. We want to try hit that 95% of the time, so we code the first threshold like this: `
  - budgetTarget: 0.95 displayName: Laggy value: 100 `This threshold requires that 95% of requests complete within 100ms.

You can name each threshold however you want. We recommend naming them how a user of the service (or how another service that uses this service) might describe the experience at a given threshold. Typically, we use names that are descriptive adjectives of the experience when the threshold is not met. When the threshold is violated, we can say that the user's experience is "Laggy".

- Some requests fall outside of that 100ms range. We want to make an allowance for that, but we also want to set other thresholds so that we know that even in its worst moments, our service is performing acceptably, and/or that its worst moments are brief.

Let's define another threshold. In the above threshold, we allow 5% of requests to run longer than 100ms. We want most of that 5%, say 80% of the remaining 5% of the queries to still return within 1/4th of a second (250ms). That means 99% of the queries return within 250ms (95% +4%). Add a threshold like this:

```
    - budgetTarget: 0.99
      displayName: Slow
      value: 250
```

This threshold requires that 99% of requests complete within 250ms.

- While that covers the bult of requests, even within the 1% of requests that we allow to exceed 250ms, the vast majority of them should complete within half a second (500ms).. Even within the 1% of requests that we allow to exceed 250ms, we want to make sure the vast majority of them complete within half a second (500ms). Add a threshold like this: `
  - budgetTarget: 0.999 displayName: Painful value: 500 `This threshold requires that 99.9% of requests complete within 500ms. Putting it all together, our SLO definition for the use cases looks like this:

```
- apiVersion: n9/v1alpha
  kind: SLO
  metadata:
    displayName: Example Latency SLO
    name: example-latency-slo
    project: default
  spec:
    alertPolicies: []
    budgetingMethod: Occurrences
    description: ""
    indicator:
      metricSource:
        name: my-prometheus-instance
        projects: default
      rawMetric:
```

```
      prometheus:
        promql: server_requestMsec{job="nginx"}
  service: my-rest-api
  thresholds:
  - budgetTarget: 0.95
    displayName: Laggy
    value: 100
  - budgetTarget: 0.99
    displayName: Slow
    value: 250
  - budgetTarget: 0.999
    displayName: Painful
    value: 500
  timeWindows:
  - count: 28
    isRolling: true
    period:
      begin: "2020-12-01T00:00:00Z"
    unit: Day
```