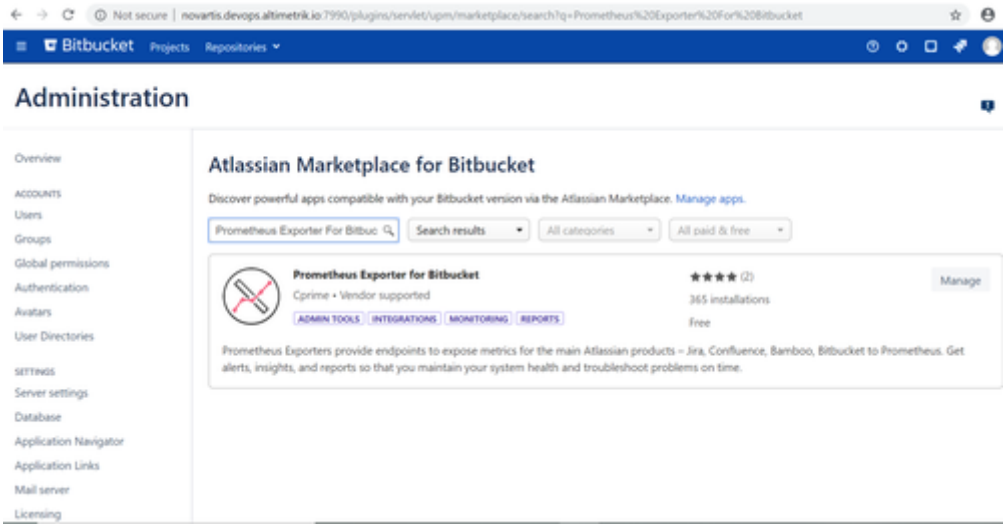
Configure Prometheus and Grafana for Bitbucket

Installation and upgrade:

- Log into your Bitbucket instance as an admin.
- Click the admin dropdown and choose **Add-ons**.The Manage add-ons screen loads.
- Click **Find new apps** or **Find new add-ons** from the left-hand side of the page.
- Locate **Prometheus Exporter for Bitbucket** via search.Results include app versions compatible with your Bitbucket instance.
- Click Install to download and install your app.

Administration- Add-ONS-Find new apps-



Usage:

After plug-in is successfully installed the following link can be used to **expose metrics: */plugins/servlet/prometheus/metrics***

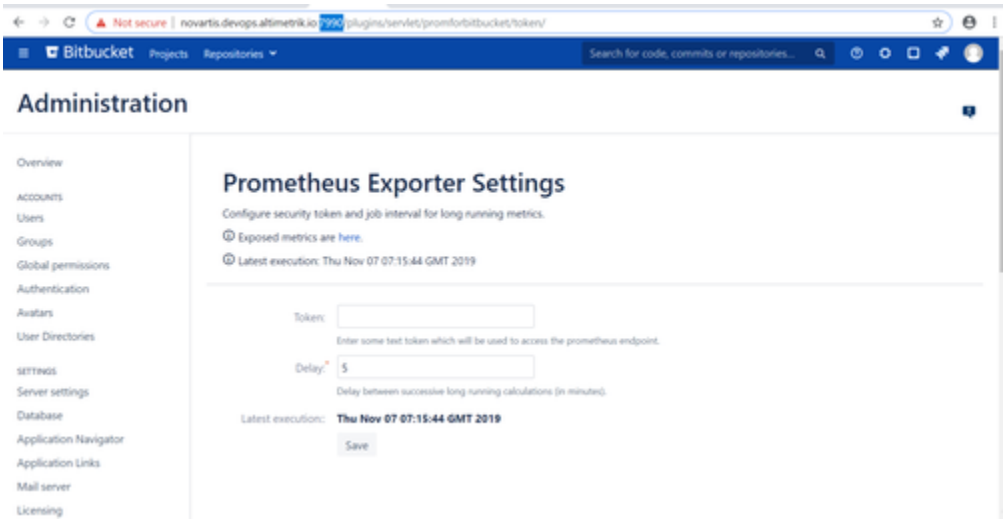
The *prometheus.yml* settings looks like:

/data/prometheus/config/prometheus.yaml

```
    scrape_configs:
    - job_name: Bitbucket-exporter
    honor_timestamps: true
    scrape_interval: 15s
    scrape_timeout: 10s
    metrics_path: /plugins/servlet/prometheus/metrics
    scheme: http
    ec2_sd_configs:
    - region: us-east-1
    access_key: AKIAXZXIY4Q4M3JKI4TO
    secret_key: 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
    profile: arn:aws:iam::536285340728:user/devplatarn
    port: 7990
    refresh_interval: 1m
    relabel_configs:
    - source_labels: [__meta_ec2_tag_Name]
    regex:'^novartis-devops-bitbucket$'
    action: keep
    - source_labels: [__meta_ec2_public_ip]
    regex: (.+)
    target_label: __address__
    replacement: novartis.devops.altimetrik.io:7990
```

Optionally you can configure secret token to allow access only by token. On Confluence administration locate and click Prometheus Exporter Settings.

In opened window specify secret token.The scrape URL will be then:
/plugins/servlet/prometheus/metrics?token=secrettoken. See, example:



Metrics:

You can access the metric

here: ***http://novartis.devops.altimetrik.io:7990/plugins/servlet/prometheus/metrics***

```
# HELP jvm_gc_collection_seconds Time spent in a given JVM garbage
collector in seconds.
# TYPE jvm_gc_collection_seconds summary
jvm_gc_collection_seconds_count{gc="G1 Young Generation",} 625.0
jvm_gc_collection_seconds_sum{gc="G1 Young Generation",} 27.934
jvm_gc_collection_seconds_count{gc="G1 Old Generation",} 0.0
jvm_gc_collection_seconds_sum{gc="G1 Old Generation",} 0.0
# HELP jvm_info JVM version info
# TYPE jvm_info gauge
jvm_info{version="1.8.0_212-b04",vendor="IcedTea",runtime="OpenJDK
Runtime Environment",} 1.0
# HELP bitbucket_success_auth_count User Success Auth Count
# TYPE bitbucket_success_auth_count counter
bitbucket_success_auth_count{username="prasad",} 41.0
bitbucket_success_auth_count{username="admin",} 121.0
bitbucket_success_auth_count{username="devlead",} 2.0
# HELP bitbucket_failed_auth_count User Failed Auth Count
# TYPE bitbucket_failed_auth_count counter
# HELP bitbucket_repo_move_count Repository Moves Count
# TYPE bitbucket_repo_move_count counter
# HELP bitbucket_repo_push_count Repository Pushes Count
# TYPE bitbucket_repo_push_count counter
bitbucket_repo_push_count{project="RTES",repository="R-mul",username="pr
asad",} 3.0
# HELP bitbucket_repo_clone_count Repository Clones Count
# TYPE bitbucket_repo_clone_count counter
bitbucket_repo_clone_count{project="RTES",repository="R-mul",username="p
```

```
rasad",} 1.0
bitbucket_repo_clone_count{project="RTES",repository="R-mul",username="a
dmin",} 32.0
# HELP bitbucket_repo_fork_count Repository Forks Count
# TYPE bitbucket_repo_fork_count counter
# HELP bitbucket_pull_request_open Opened Pull Requests Count
# TYPE bitbucket_pull_request_open counter
# HELP bitbucket_pull_request_merge Merged Pull Requests Count
# TYPE bitbucket_pull_request_merge counter
# HELP bitbucket_pull_request_decline Declined Pull Requests Count
# TYPE bitbucket_pull_request_decline counter
# HELP bitbucket_maintenance_expiry_days_gauge Maintenance Expiry Days
Gauge
# TYPE bitbucket_maintenance_expiry_days_gauge gauge
bitbucket_maintenance_expiry_days_gauge 6.0
# HELP bitbucket_license_expiry_days_gauge License Expiry Days Gauge
# TYPE bitbucket_license_expiry_days_gauge gauge
bitbucket_license_expiry_days_gauge 6.0
# HELP bitbucket_active_users_gauge Active Users Gauge
# TYPE bitbucket_active_users_gauge gauge
bitbucket_active_users_gauge 8.0
# HELP bitbucket_allowed_users_gauge Maximum Allowed Users
# TYPE bitbucket_allowed_users_gauge gauge
bitbucket_allowed_users_gauge -1.0
# HELP bitbucket_total_projects_gauge Total Projects Gauge
# TYPE bitbucket_total_projects_gauge gauge
bitbucket_total_projects_gauge 11.0
# HELP bitbucket_total_repositories_gauge Total Repositories Gauge
# TYPE bitbucket_total_repositories_gauge gauge
bitbucket_total_repositories_gauge 19.0
# HELP bitbucket_total_pull_requests_gauge Total Pull Requests Gauge
# TYPE bitbucket_total_pull_requests_gauge gauge
bitbucket_total_pull_requests_gauge 4.0
# HELP bitbucket_plugin_installed Plugin Installed Count
# TYPE bitbucket_plugin_installed counter
# HELP bitbucket_plugin_uninstalled Plugin Uninstalled Count
# TYPE bitbucket_plugin_uninstalled counter
# HELP bitbucket_plugin_enabled Plugin Enabled Count
# TYPE bitbucket_plugin_enabled counter
# HELP bitbucket_plugin_disabled Plugin Disabled Count
# TYPE bitbucket_plugin_disabled counter
# HELP process_cpu_seconds_total Total user and system CPU time spent in
seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 74202.46
# HELP process_start_time_seconds Start time of the process since unix
epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.573148220678E9
# HELP process_open_fds Number of open file descriptors.
```

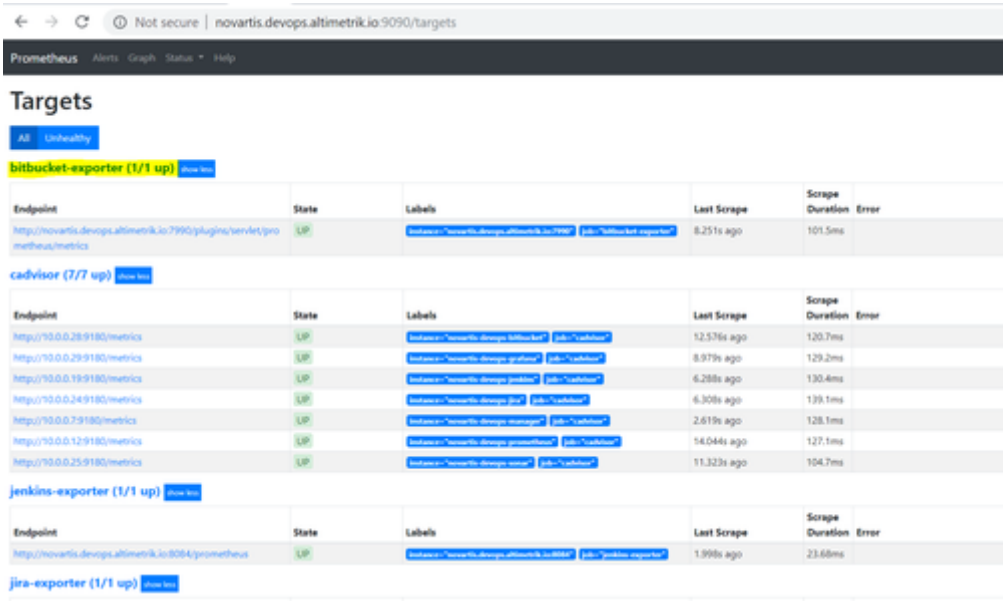
```
# TYPE process_open_fds gauge
process_open_fds 723.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 1048576.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 3.236888576E9
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 1.555955712E9
# HELP jvm_memory_pool_allocated_bytes_total Total bytes allocated in a
given JVM memory pool. Only updated after GC, not continuously.
# TYPE jvm_memory_pool_allocated_bytes_total counter
# HELP jvm_classes_loaded The number of classes that are currently
loaded in the JVM
# TYPE jvm_classes_loaded gauge
jvm_classes_loaded 48831.0
# HELP jvm_classes_loaded_total The total number of classes that have
been loaded since the JVM has started execution
# TYPE jvm_classes_loaded_total counter
jvm_classes_loaded_total 48853.0
# HELP jvm_classes_unloaded_total The total number of classes that have
been unloaded since the JVM has started execution
# TYPE jvm_classes_unloaded_total counter
jvm_classes_unloaded_total 22.0
# HELP jvm_buffer_pool_used_bytes Used bytes of a given JVM buffer pool.
# TYPE jvm_buffer_pool_used_bytes gauge
jvm_buffer_pool_used_bytes{pool="direct",} 326405.0
jvm_buffer_pool_used_bytes{pool="mapped",} 0.0
# HELP jvm_buffer_pool_capacity_bytes Bytes capacity of a given JVM
buffer pool.
# TYPE jvm_buffer_pool_capacity_bytes gauge
jvm_buffer_pool_capacity_bytes{pool="direct",} 326405.0
jvm_buffer_pool_capacity_bytes{pool="mapped",} 0.0
# HELP jvm_buffer_pool_used_buffers Used buffers of a given JVM buffer
pool.
# TYPE jvm_buffer_pool_used_buffers gauge
jvm_buffer_pool_used_buffers{pool="direct",} 35.0
jvm_buffer_pool_used_buffers{pool="mapped",} 0.0
# HELP jvm_threads_current Current thread count of a JVM
# TYPE jvm_threads_current gauge
jvm_threads_current 203.0
# HELP jvm_threads_daemon Daemon thread count of a JVM
# TYPE jvm_threads_daemon gauge
jvm_threads_daemon 178.0
# HELP jvm_threads_peak Peak thread count of a JVM
# TYPE jvm_threads_peak gauge
jvm_threads_peak 208.0
# HELP jvm_threads_started_total Started thread count of a JVM
```

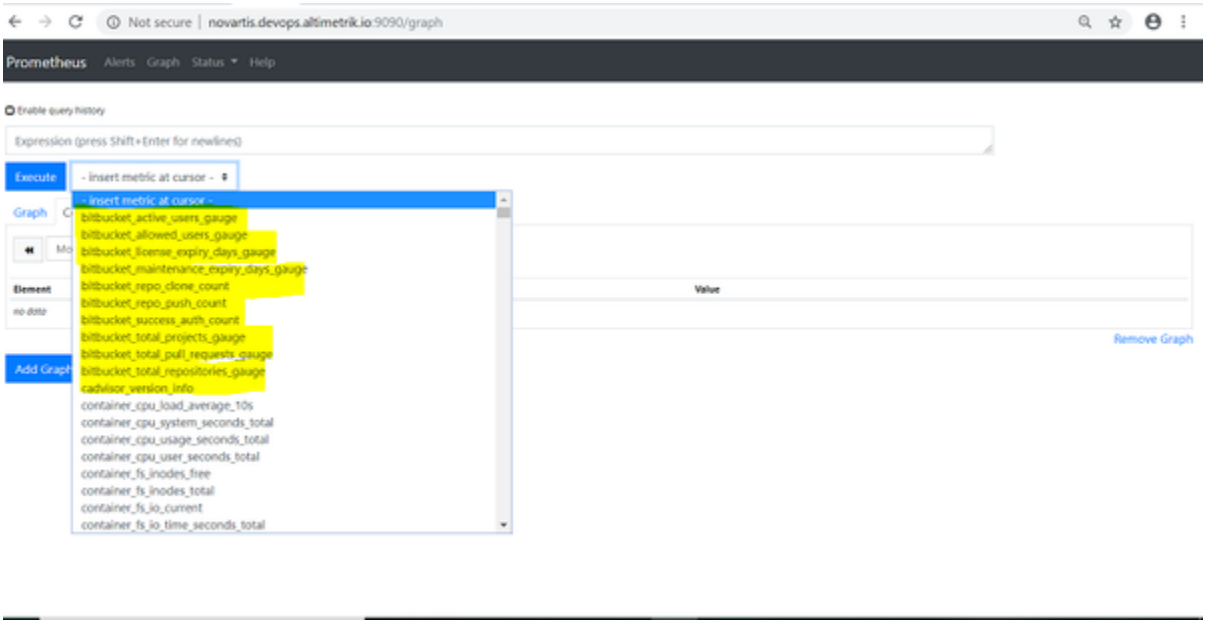
```
# TYPE jvm_threads_started_total counter
jvm_threads_started_total 856.0
# HELP jvm_threads_deadlocked Cycles of JVM-threads that are in deadlock
waiting to acquire object monitors or ownable synchronizers
# TYPE jvm_threads_deadlocked gauge
jvm_threads_deadlocked 0.0
# HELP jvm_threads_deadlocked_monitor Cycles of JVM-threads that are in
deadlock waiting to acquire object monitors
# TYPE jvm_threads_deadlocked_monitor gauge
jvm_threads_deadlocked_monitor 0.0
# HELP jvm_threads_state Current count of threads by state
# TYPE jvm_threads_state gauge
jvm_threads_state{state="NEW",} 0.0
jvm_threads_state{state="BLOCKED",} 0.0
jvm_threads_state{state="TERMINATED",} 0.0
jvm_threads_state{state="WAITING",} 108.0
jvm_threads_state{state="TIMED_WAITING",} 38.0
jvm_threads_state{state="RUNNABLE",} 57.0
# HELP jvm_memory_bytes_used Used bytes of a given JVM memory area.
# TYPE jvm_memory_bytes_used gauge
jvm_memory_bytes_used{area="heap",} 5.30664224E8
jvm_memory_bytes_used{area="nonheap",} 3.92174832E8
# HELP jvm_memory_bytes_committed Committed (bytes) of a given JVM
memory area.
# TYPE jvm_memory_bytes_committed gauge
jvm_memory_bytes_committed{area="heap",} 9.35329792E8
jvm_memory_bytes_committed{area="nonheap",} 4.170752E8
# HELP jvm_memory_bytes_max Max (bytes) of a given JVM memory area.
# TYPE jvm_memory_bytes_max gauge
jvm_memory_bytes_max{area="heap",} 1.073741824E9
jvm_memory_bytes_max{area="nonheap",} -1.0
# HELP jvm_memory_bytes_init Initial bytes of a given JVM memory area.
# TYPE jvm_memory_bytes_init gauge
jvm_memory_bytes_init{area="heap",} 5.36870912E8
jvm_memory_bytes_init{area="nonheap",} 2555904.0
# HELP jvm_memory_pool_bytes_used Used bytes of a given JVM memory pool.
# TYPE jvm_memory_pool_bytes_used gauge
jvm_memory_pool_bytes_used{pool="Code Cache",} 1.14216448E8
jvm_memory_pool_bytes_used{pool="Metaspace",} 2.46104128E8
jvm_memory_pool_bytes_used{pool="Compressed Class Space",} 3.1854256E7
jvm_memory_pool_bytes_used{pool="G1 Eden Space",} 1.49946368E8
jvm_memory_pool_bytes_used{pool="G1 Survivor Space",} 2097152.0
jvm_memory_pool_bytes_used{pool="G1 Old Gen",} 3.78620704E8
# HELP jvm_memory_pool_bytes_committed Committed bytes of a given JVM
memory pool.
# TYPE jvm_memory_pool_bytes_committed gauge
jvm_memory_pool_bytes_committed{pool="Code Cache",} 1.15146752E8
jvm_memory_pool_bytes_committed{pool="Metaspace",} 2.6550272E8
jvm_memory_pool_bytes_committed{pool="Compressed Class Space",}
3.6425728E7
```

```
jvm_memory_pool_bytes_committed{pool="G1 Eden Space",} 4.80247808E8
jvm_memory_pool_bytes_committed{pool="G1 Survivor Space",} 2097152.0
jvm_memory_pool_bytes_committed{pool="G1 Old Gen",} 4.52984832E8
# HELP jvm_memory_pool_bytes_max Max bytes of a given JVM memory pool.
# TYPE jvm_memory_pool_bytes_max gauge
jvm_memory_pool_bytes_max{pool="Code Cache",} 2.5165824E8
jvm_memory_pool_bytes_max{pool="Metaspace",} -1.0
jvm_memory_pool_bytes_max{pool="Compressed Class Space",} 1.073741824E9
jvm_memory_pool_bytes_max{pool="G1 Eden Space",} -1.0
jvm_memory_pool_bytes_max{pool="G1 Survivor Space",} -1.0
jvm_memory_pool_bytes_max{pool="G1 Old Gen",} 1.073741824E9
# HELP jvm_memory_pool_bytes_init Initial bytes of a given JVM memory
pool.
# TYPE jvm_memory_pool_bytes_init gauge
jvm_memory_pool_bytes_init{pool="Code Cache",} 2555904.0
jvm_memory_pool_bytes_init{pool="Metaspace",} 0.0
jvm_memory_pool_bytes_init{pool="Compressed Class Space",} 0.0
```

```
jvm_memory_pool_bytes_init{pool="G1 Eden Space",} 2.8311552E7
jvm_memory_pool_bytes_init{pool="G1 Survivor Space",} 0.0
jvm_memory_pool_bytes_init{pool="G1 Old Gen",} 5.0855936E8
```

Installation and setup is finished. You can collect metrics. Prometheus will be available at http://novartis.devops.altimetrik.io:9090/

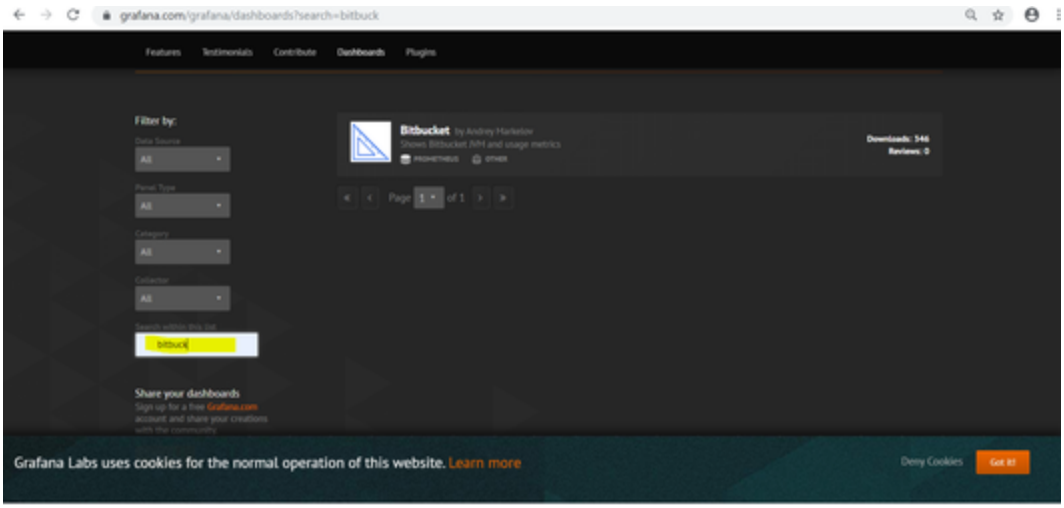




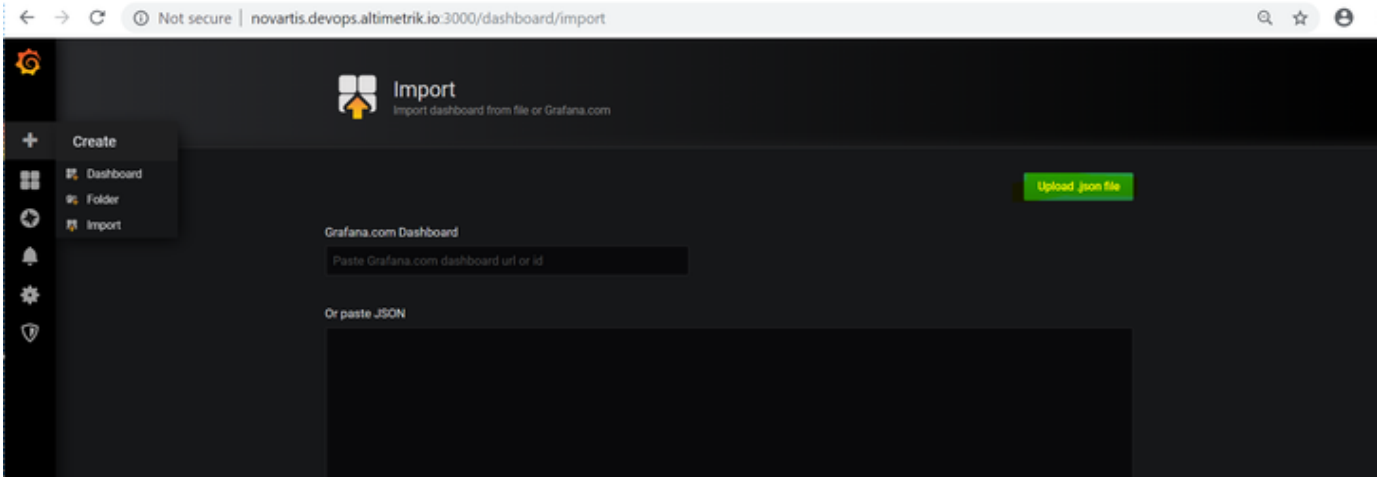
Grafana Configuration

We will use Grafana to visualize metrics stored in Prometheus. There are a couple of example dashboards in the official site https://grafana.com/grafana/dashboards

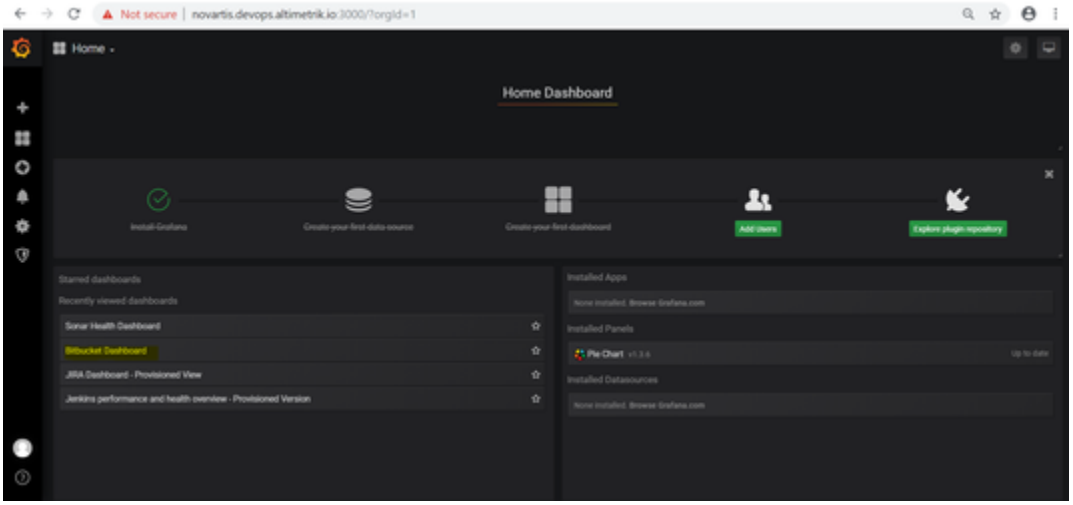
Download the latest release of Grafana for your platform, then extract it:



Go to Grafana interface at http://novartis.devops.altimetrik.io:3000/dashboard/import

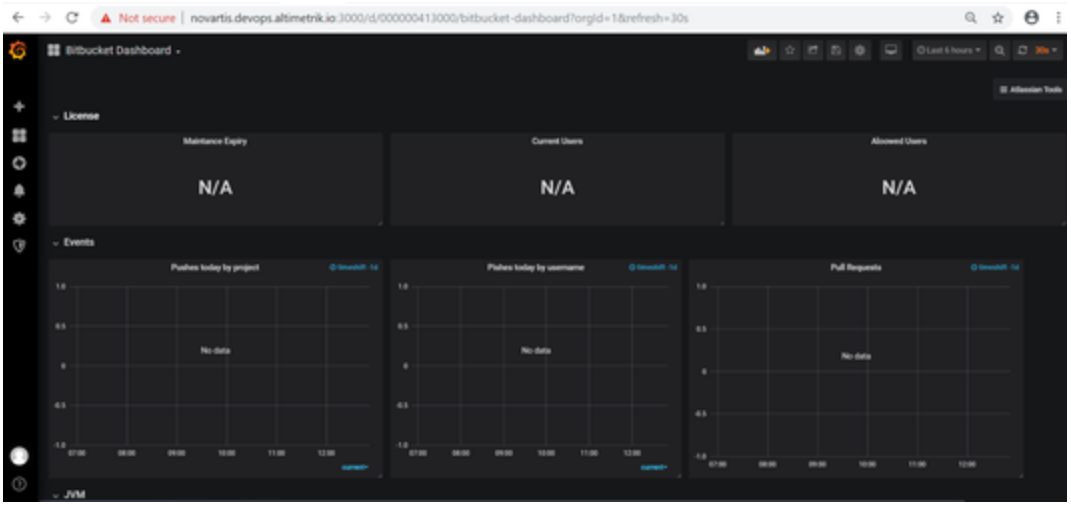


Then, you can add a new Dashboard that will get the data from the previous created Data-Source.



After a couple of minutes you will be able to view your metrics on the Dashboard. You can also add new panels to the Dashboard.

You can also build one Dashboard to tack all your Atlassian tools in one single Dashboard.



Eyes wide open (source)

Keeping an eye on application performance is key for maintaining high productivity, healthy workflows, and happy team members (and customers).