Tableau – VizAlerts Installation Guide

Contents

[What is VizAlerts? 2](#_Toc432417574)

[What does it do? 2](#_Toc432417575)

[How does it work? 2](#_Toc432417576)

[Prerequisites 3](#_Toc432417577)

[Tableau Server 3](#_Toc432417578)

[Windows Host Machine 3](#_Toc432417579)

[SMTP (Mail) Server 3](#_Toc432417580)

[Setup 4](#_Toc432417581)

[Configure Tableau Server 4](#_Toc432417582)

[Trusted Tickets 4](#_Toc432417583)

[Repository Access 4](#_Toc432417584)

[Restart 4](#_Toc432417585)

[Custom Subscription Schedules 4](#_Toc432417586)

[Install Python & Required Modules 6](#_Toc432417587)

[Configure VizAlerts 7](#_Toc432417588)

[Testing 8](#_Toc432417589)

[Final Steps 9](#_Toc432417590)

[Set up a Scheduled Task 9](#_Toc432417591)

[Helper Datasource 11](#_Toc432417592)

[FAQ 11](#_Toc432417593)

[Common Errors 12](#_Toc432417594)

[Appendix A 12](#_Toc432417595)

# What is VizAlerts?

VizAlerts is an email automation platform intended to seamlessly integrate with Tableau Server. The idea behind it is that anyone should be able to easily build, share, and customize pretty much any email automation based on their own Tableau Server viz data.

In its current form, VizAlerts exists simply as a Python script that is set up by a system administrator to run at frequent and regular intervals. All of the information it needs to enable data-driven email alerting is derived from the PostgreSQL repository and published views of the Tableau Server it is set to run against.

# What does it do?

VizAlerts is extremely flexible and allows for a wide variety of email alerts. Here are some examples that illustrate the breadth of use cases it has:

* Alert me when I’ve achieved my sales quota
* Alert my team when I’ve achieved my sales quota
* Alert each individual on my team when they’ve achieved their sales quota, and CC the sales team alias
* Alert the entire Sales team when the collective Sales revenue target has been reached, and include contextual information on past quarters by rendering a dashboard that displays that information
* Customize an alert someone else made to let them know when they’ve achieved an all-time sales record for themselves in order make it relevant to my own performance and desired alert threshold
* Build an alert for myself, but allow anyone else to subscribe to it and be alerted in the same way that I am

# How does it work?

While the details on how to *set up* alerts will be left in the User Guide, it’s important for Administrators to know how things are working behind the scenes.

The general flow of a single execution of the vizalerts.py script goes like this:

1. Connect to the PostgreSQL repository database of Tableau Server, and query it for a list of Views subscribed to on specially configured, disabled Schedules
2. Compare Schedule information to last runtime information stored in a local text file—based on this, determine which Views are due for alert testing
3. For each View found due for testing:
   1. Generate a Trusted Ticket on behalf of the Subscriber of the View
   2. Redeem the Trusted Ticket to export the CSV data for the View, impersonating the Subscriber
   3. If one or more rows are found in the CSV:
      1. For a “Simple Alert”, generate a new Trusted Ticket, export the PNG of the View, and email it to the Subscriber
      2. For an “Advanced Alert”, iterate through each row of the CSV, sending emails as instructed by the data itself

Prerequisites

## Tableau Server

The Tableau Server instance that you wish to run VizAlerts against must fulfill the following requirements:

* Must be v8.2.5 or higher
* The [readonly user](http://onlinehelp.tableau.com/current/server/en-us/adminview_postgres_access.htm) must be granted Repository database access
* [Subscriptions](http://onlinehelp.tableau.com/current/server/en-us/subscribe.htm) must be enabled
* The host you plan to run VizAlerts from must have its IP address listed as a [Trusted Host](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth_trustIP.htm)
* If it wasn’t already obvious, you need to be a System Administrator on Server to set all this up.

## Windows Host Machine

This is where VizAlerts will be run from, which means that this machine must be continually up and running for VizAlerts to function. This can be one of the Tableau Server hosts if desired, but it doesn’t have to be. It must have the following properties:

* Static IP address
* Always running
* Within same domain as Tableau Server
* You must have administrative rights to it
* Should **not** need to have much processing power as heavy work is offloaded to Tableau Server

## SMTP (Mail) Server

VizAlerts needs to point to a mail server to send email. This can simply be the same server you used when you set up Tableau Server for subscriptions. If your mail server is set up to support SSL encryption, that is ideal, but it’s not required.

# Setup

You’ve got everything you need, now let’s get this thing running!

## Configure Tableau Server

Making any of these configuration changes requires a restart of Tableau Server, so if this is being done on a live / production server, make sure to do this during a maintenance window.

### Trusted Tickets

VizAlerts uses [Trusted Authentication](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth.htm) to impersonate users and obtain access to Tableau Server views in CSV and PNG format. To grant it this access, run the following command at a command prompt on the Primary host of Tableau Server:

tabadmin set wgserver.trusted\_hosts <HOSTNAME OF VIZALERTS HOST>

### Repository Access

The Tableau Server repository database contains information VizAlerts needs to function. Grant it access by enabling the [readonly user](http://onlinehelp.tableau.com/current/server/en-us/adminview_postgres_access.htm):

tabadmin dbpass --username readonly <YOUR PASSWORD>

### Restart

Once you have finished the above steps, you must save the configuration and restart Tableau Server. When you’re ready to do this, run the following commands in the command prompt:

tabadmin configure

tabadmin restart

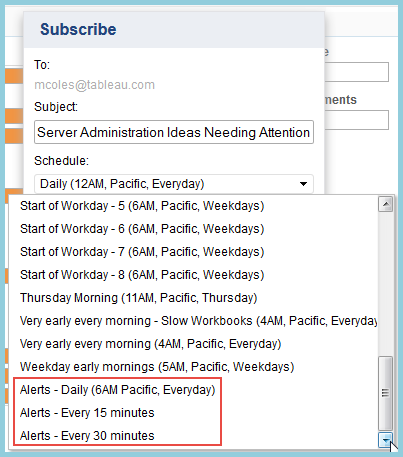
### Custom Subscription Schedules

A key component that allows VizAlerts to work in the intuitive way that it does is that users who wish to schedule an alert are able to subscribe to them on *disabled* Subscriptions schedules. These are schedules that you must create in Tableau Server, then manually disable so that no subscriptions are ever delivered for them. Since the data for who subscribed to what views *on* these specific schedules exists in the PostgreSQL repository, VizAlerts can use this information to tell itself when it is appropriate to test those views for an alert condition.

You can create as many schedules as you like, on whatever intervals you like. The important bit behind the schedules is the **naming convention** that you use, because this is how VizAlerts knows which schedules to consider “alert” schedules that it needs to pay attention to. I recommend naming them like this:

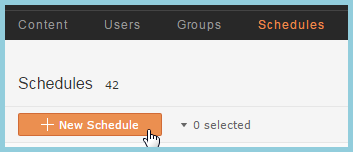
Аlerts – [frequency]

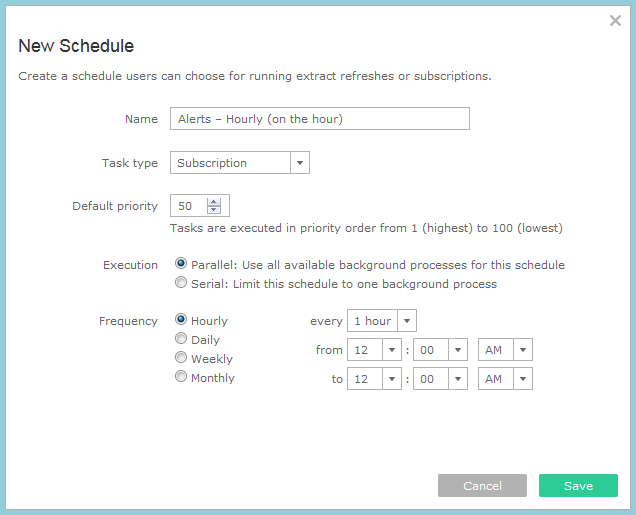
**Copy and paste that** when you create your schedules—the first letter is actually the Cyrillic letter A, which will cause your Alerts schedules to be sorted at the bottom of the list when someone goes to subscribe. This can help users avoid subscribing to them by mistake when they only mean to set up a standard subscription:

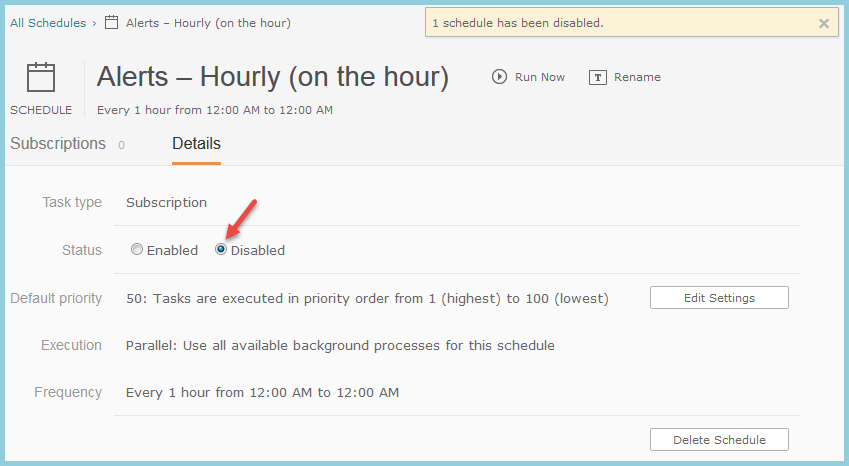


Note that you must have at least one **enabled** Subscriptions schedule for anyone to subscribe to a viz on Tableau Server, so if you have just enabled Subscriptions for the first time, you’ll also need to create a single non-Alert schedule that isn’t disabled.

Create your new schedules like so:

­­­­­­





## Install Python & Required Modules

1. On the Windows host you want to run VizAlerts from, download and install Python 2.7. This can be done in multiple ways, but we suggest this MSI installer: <https://www.python.org/ftp/python/2.7.9/python-2.7.9.msi>
2. Add ";C:\Python27\;C:\Python27\Scripts\" to your Path environment variable (assuming you chose the installation defaults when installing Python)
3. Install the following Python modules:
   1. [PyYAML](http://pyyaml.org/) (recommended: <http://pyyaml.org/download/pyyaml/PyYAML-3.11.win32-py2.7.exe> )
   2. [psycopg2](http://www.stickpeople.com/projects/python/win-psycopg/) (recommended Windows port: <http://www.stickpeople.com/projects/python/win-psycopg/2.6.0/psycopg2-2.6.0.win32-py2.7-pg9.4.1-release.exe> )
   3. The final two packages, [requests](http://docs.python-requests.org/en/latest/user/install/#install) and [requests\_ntlm](https://github.com/requests/requests-ntlm/), are best installed by opening a *new* command prompt and running the following commands:  
        
      *pip install requests  
      pip install requests\_ntlm*  
      If your computer does not have access to the Internet, see [Appendix A](#_Appendix_A).

## Configure VizAlerts

Now that Python is installed, we can configure VizAlerts. Unzip the VizAlerts.zip file to a folder of your choosing. For the purposes of this manual, we’ll assume the files were extracted to C:\VizAlerts.

1. The next task is to give VizAlerts all the information it needs to connect to our Tableau Server instance. Open the file C:\VizAlerts\config\vizalerts.yaml in a text editor. Each of the configuration settings in that file are commented to explain what they do, but we’ll go over the most important ones here:

|  |  |
| --- | --- |
| **Email Settings** |  |
| smtp.serv | This is the name of your SMTP server. |
| smtp.address.from | The email address you wish all email alerts to be sent from. **Note** that for Advanced Alerts, this is used only if the author did not specify their own “from” address in their viz. |
| smtp.address.to | When an alert fails to run, failure details will be sent to this address along with the Subscriber, so it makes the most sense to use your own address or Admin distribution list here. |
| smtp.alloweddomains | You probably don’t want people sending alerts out to @aol.com addresses, so this is a whitelist of domains that VizAlerts can send mail to. Any alert that attempts to send mail to any other domain will fail. This list must be enclosed in square brackets, with multiple values being separated by commas. |
| smtp.ssl | When true, VizAlerts will attempt to use SSL for email encryption (which your SMTP server must support). If you do not wish to use encryption, leave it “false”. |
| smtp.user | Username for the account used to connect to your SMTP server. If no authentication is need, leave it “null” |
| smtp.password | Password for the account used to connect to your SMTP server. If no authentication is need, leave it “null”. The password must be enclosed in single quotes.  If desired, this value can be a valid path to a .txt file containing the password, e.g. 'c:\users\mcoles\password.txt', rather than the password itself. |

|  |  |
| --- | --- |
| **Tableau Server Settings** |  |
| server | Name of the Tableau Server you wish to run this instance of VizAlerts against. |
| server.version | Major version of the Tableau Server you are running VizAlerts against (this must be 8 or 9) |
| server.user | This is ANY user licensed in Tableau Server--it does not need to be an Admin, as it is only used in authenticating over HTTP.   * If you are using Active Directory authentication, prepend the domain name in front of the username, e.g. “tableau.com\mcoles” * If you are using Local Authentication, simply supply the username, e.g., “mcoles” |
| server.ssl | When set to true, use SSL to connect to Tableau Server (recommended if you have enabled SSL). |

|  |  |
| --- | --- |
| **Tableau Server Repository Settings** |  |
| db.host | This is the host name of the Tableau Server worker responsible for running the Repository process. If you have two, it’s recommended you pick the Secondary, though it doesn’t matter that much. |
| db.pw | Password for the user you connect to the Repository database with. It should be the password you set (or plan to set) when you enabled the “readonly” user access. The password should be enclosed in single quotes.  If desired, this value can be a valid path to a .txt file containing the password, e.g. 'c:\users\mcoles\password.txt', rather than the password itself. |
| db.query | This is the query that is run against the Repository database to obtain the set of Alerts to run each time VizAlerts is executed.  The only **important** change here is that you must ensure the pattern in this line matches your Schedule names. **If you followed the naming convention** in this guide, you do not need to alter this. The default pattern is looking for “[any single character]lerts[any number of any characters]”:  -- ######################################################  -- ######## Editable Filters #########  -- ######################################################  AND LOWER(sch.name) LIKE '\_lerts%’  **IMPORTANT:** The “editable filters” block is the **only** part of this query that you should modify. You can edit the other adjacent lines to only allow specific users, Projects, or Sites for Alerts. |

## Testing

Whew! All that was lots of fun, but let’s get to the good stuff and test this thing to see if we did everything right. Run the following from a command prompt on the Windows host you set VizAlerts up on:

cd C:\VizAlerts

python C:\VizAlerts\vizalerts.py

It should have successfully generated a Trusted Ticket, queried the PostgreSQL database in Tableau Server, then realized there was nothing to do and quit without error. If it didn’t, please see the [Common Errors](#_Common_Errors:) section.

Now, for a more extensive test. Subscribe to any Tableau Server View on an Alerts schedule that you set up (pick a view that renders in less than 10 seconds or so). I recommend subscribing on an Alerts schedule that runs every 15 minutes for this test, even if you just remove it afterward. After you subscribe, run the command again:

python C:\VizAlerts\vizalerts.py

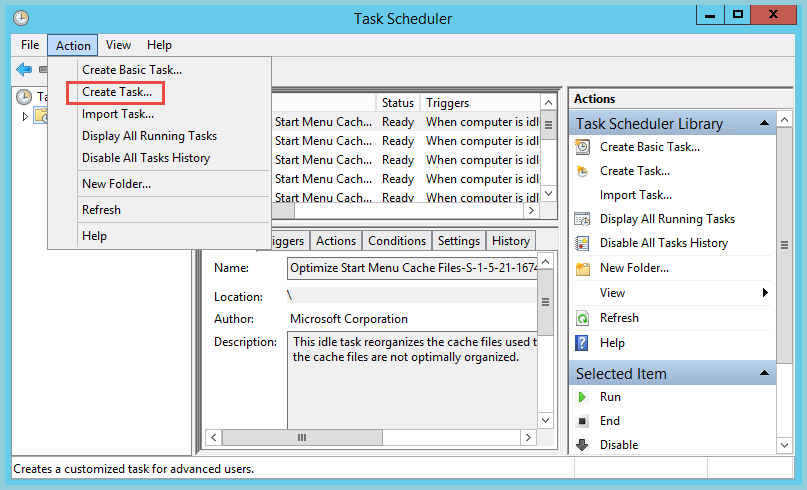
Now, wait 15 minutes, then run the same command again. If data is present in the viz, you should receive an email! If not, you shouldn’t. Simple as that!

## Final Steps

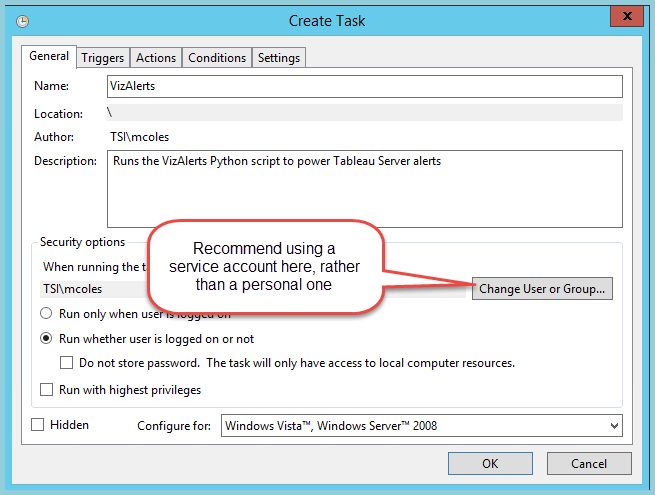
### Set up a Scheduled Task

The last step, now that everything is working as expected, is to automate this so that VizAlerts can run regularly when it is supposed to. To do this, we need to set up a Scheduled Task on the Windows host that VizAlerts runs from, which will run this for us on a regular basis.

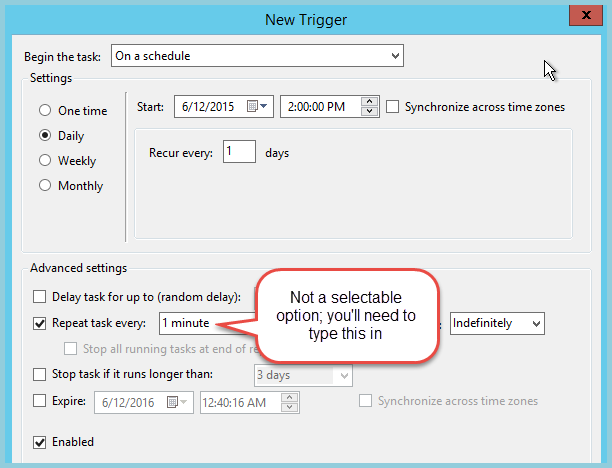
First, let’s create a new Task:



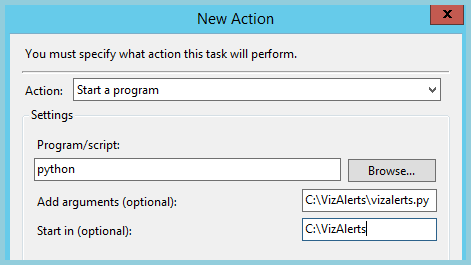
Fill in the name and description. Make sure it will run whether the user is logged in or not. The task should be set up to run under a service account rather than a personal one, if possible. This account must have full control permissions on the VizAlerts files, and if you specified text files instead of passwords in the vizalerts.yaml config file, the account will need rights to read those files.



Set up the Trigger (when will it run?). I strongly recommend running this every **1 minute**, as this will keep alerts executing on time, and the vast majority of executions will be quick checks that don’t actually do any work:



Set the Action on the Task (what will it do?)



### Helper Datasource

Last, but not least, publish up the VizAlerts.tdsx datasource that came in the VizAlerts.zip file to Tableau Server, and grant permissions to anyone you wish to have an easy way to create Advanced Alerts. This datasource simply gives users a shortcut to creating them (see the User Guide for more details).

### 

# FAQ

* **How many alerts can be run at once?**Alerts are processed serially (one at a time) each time they are checked on Schedule. Alerts schedule for 6AM will begin being checked at 6AM, so if several long-running alerts are checked at that time, subsequent alerts will be checked sometime after 6AM. This can be exacerbated if long-running alerts aren’t removed from the pipeline, and/or reasonable timeouts set in the config file.  
    
  **NOTE** that it *is* possible to run multiple alerts in parallel, by scheduling additional Scheduled Task executions of vizalerts.py, each of which point at separate config files, and ops/log/temp folders. If you do this, you will also need to edit the SQL query within each vizalerts.yaml config file so that each scheduled task focuses on an isolated set of alerts for say, a Project, or User, or even just picking Subscriptions with IDs that are even / odd, or what have you.
* **Does VizAlerts use a database to log information about what it has done?**No, not in its current state, though this is the next logical progression for it. Currently it logs information into text files only.

# Common Errors

* **Failed with unknown protocol**
  + This likely means that you’ve enabled SSL in the vizalerts.yml, but haven’t set Tableau Server up for it. See [this portion](http://onlinehelp.tableau.com/current/server/en-us/ssl_config.htm) of the online help on how to do so.
* **Parse errors**
  + Generally this means that some bad character or formatting issue was introduced to the vizalerts.yml file (typically a tab character—replace them with spaces!). I recommend using this [online YAML validator](http://codebeautify.org/yaml-validator) to determine where the problem is (make sure to remove your passwords first!).
* **Postgres connection failure**
  + Ensure the [readonly user](http://onlinehelp.tableau.com/current/server/en-us/adminview_postgres_access.htm) is set up for PostgreSQL repository access, and that the password you’ve specified is correct.
* **Trusted ticket failure**
  + Check to ensure your trusted tickets were [configured properly](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth_trustIP.htm). If things are still not working, try [this article](http://kb.tableau.com/articles/knowledgebase/testing-trusted-authentication) to test them further.
  + A “-1” result could be due to several possible issues. Please see [this article](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth_trouble_1return.htm) if you’re seeing this error.
* **Unable to export … as CSV**
  + This means that the attempt to export the view data for an alert to a CSV file failed, either because of internal errors, or because it took longer than the timeout you’ve set in the config file. If the view can be rendered successfully in your browser, it may simply be taking too long. Increasing the timeout settings may help with this, but a better solution is to try and optimize the viz to render more quickly. By default, the settings use stricter timeouts on more frequently-run alerts, as it’s assumed they’ll have more opportunities to be retried.

# Appendix A

1. To install the Python modules on an Internet-enabled machine, download each package to a folder of your choosing. This functions as basically an offline package repository:

*pip install --download c:\mypythonpackages requests\_ntlm*

1. Copy the folder to your offline machine (I'm assuming here that it's copied to the same path).
2. On your offline machine, install the package from the newly copied folder:

*pip install --no-index --find-links file:c:\mypythonpackages requests\_ntlm*