Merkator

Continuous Deployment with ArcGIS Pro, Publish After GIT Commit

Joël Hempenius
Merkator GeoSpatial-IT Solutions

Continuous delivery, Continuous deployment

Merkator

- Automated builds
  - Version control
  - Automated builds

- Continuous delivery
  - Version control
  - Automated builds
  - Automated tests
  - *Manual users tests*
  - Automated release

- Continuous deployment
  - *Automated users tests*

Are you ready?

- GIS specialists with developer skills
- Test automation skills
- Management support
- Business support

ArcGIS Enterprise

Portal for Arcgis

User Content
Webapps
Webmaps
Hosted
Featureservices
Other data

Managed content
Webapps
Webmaps
Mapservices
Featureservices
Geoprocessingservices

Custom GIS applications

Other corporate applications

Arcgis Server
Hosting Server

Arcgis Server
Federated Server

Arcgis Datastore

Enterprise GDB

The Alliander Case

- Gas and electric utility company in The Netherlands
  - 6 million clients
  - 35.000 km pipes
  - 85.000 km cables

- Smallworld as System of Record

- ArcGIS Server, Portal and Geocortex as System of Engagement
  - Portal DTAP
  - ArcGIS Server 24/7 DTAP
    - Production and failover backup
  - ArcGIS Server DTAP
  - 12 ArcGIS Server Sites, 21 ArcGIS Server machines
  - 200+ Mapservices

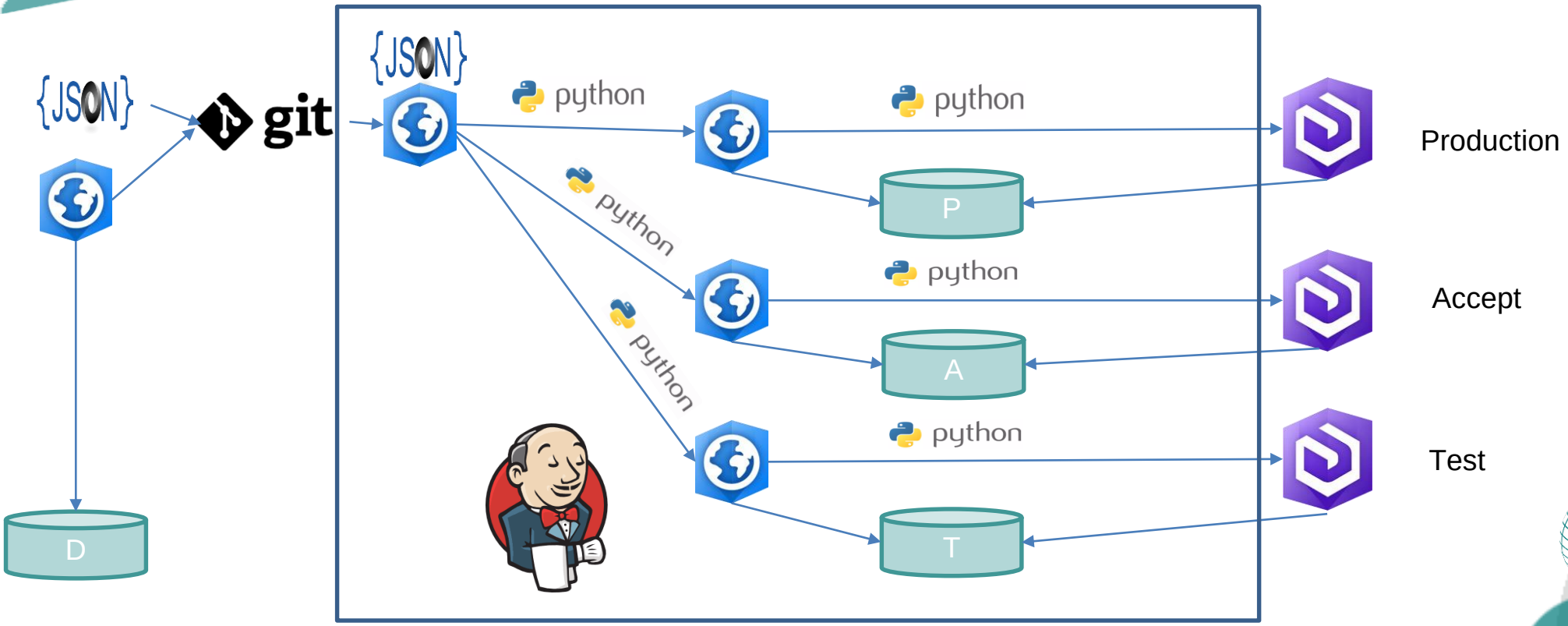
The Alliander Case

- Change from file share for MXD’s to GIT Version Control
- Change from manual Mapservice publishing to Arcpy Scripted publishing
- Change from manual testing to automated testing with Robot Framework
- More frequent performance testing with Jmeter
- Agile&Scrum&DevOps with frequent releases

Components

- GIT
  - Version control of ArcMap or ArcGIS Pro Document and JSON configuration files
- Arcpy and Arcgis API for Python scripts
- Robot framework
- JMeter
  - Performance test scripts for MapServices and FeatureServices
- Jenkins Build Server
  - GIT integration
  - MapService publishing jobs
  - JMeter Performance test jobs
  - Build pipelines
- Windows Server 2012R2 4GB RAM, 2VCPU

From local Pro project to Enterprise Mapservice



GIT

- MXD and APRX are not GIT friendly
  - Small and incremental commits
  - Commit with messages describing what’s changed

- Protect your GIT repositories!
  - Private repository
  - On premise GIT server

Arcpy and ArcGIS API for Python

- Arcpy
  - Replacing datasources
  - Create Service draft file
  - Create Service definition file
  - *Validate and register database connection files (.sde) with server*
  - Publish to Arcgis Server

- ArcGIS API for Python
  - Validate and register database connection files (.sde) with server
  - *Publish to Arcgis Server*
  - Update service properties
  - Update sharing properties

Mapservice configuration JSON

- Generic configuration items
- Server specific configuration items
- Server specific database configuration items


```
{
  "action": "publishSD",
  "name": "TestQueryLayer",
  "serviceType": "MapServer",
  "aprx": "TestQueryLayer.aprx",
  "summary": "My Mapservice summary",
  "description": "My Mapservice description",
  "tags": "Tag",
  "credits":"Copyright Joel",
  "uselimitations":"Testdoeleinden",
  "serverFolder": "Demo",
  "portalFolder":"Demo",
  "capabilities": "Map,Query,Data",
  "maxIdleTime": 1800,
  "antialiasingMode": "None",
  "textAntialiasingMode":"Force",
  "maxRecordCount":1000,
  "recycleInterval": 24,
  "minInstancesPerNode": 2,
  "maxInstancesPerNode": 2,
  "servers":{
    "EDN_T": {
      "minInstancesPerNode": 1,
      "maxInstancesPerNode": 1,
      "serverFolder": "Test",
      "datasources": [
        {"connection_info": {"authentication_mode": "DBMS",
                  "database": "gis_t",
                  "db_connection_properties": "                ",
                  "dbclient": "postgresql",
                  "instance": "sde:postgresql:                ",
                  "password": "",
                  "server": "my.database.local",
                  "user": "sde_schema_user",
                  "version": "sde.DEFAULT"},
          "database": "gis_t",
          "schema":"sde_ov",
          "layers":["polygon_feature_class","line_feature_class"],
          "tables":["data_table"],
          "workspace_factory": "SDE",
```

Replace datasources

```
firstmap = aprx.listMaps("*")[0]
try:
    for lyr in firstmap.listLayers("*"):
        if lyr.supports('DATASOURCE'):
            oldConnectionProperties = lyr.connectionProperties
            layername = oldConnectionProperties['dataset']
            databasetable = oldConnectionProperties['dataset'].split('.')[-1]
            replaced = False
            for datasource in datasources:
                layers = datasource['layers']
                #check if database table is listed in layers list
                if layers == "*" or databasetable in layers:
                    logging.info( "Update datasource: " + layername)
                    if 'connection_info' in datasource:
                        validate = True
                        if 'validate' in datasource:
                            validate = datasource['validate'] == "true"
                        newDatasetName = datasource['database'] + '.' + datasource['schema'] + '.' + databasetable if 'database' in datasource else datasource['sc
                        newConnectionProperties = {'connection_info': datasource['connection_info'], 'workspace_factory': datasource['workspace_factory'], 'datase
                        #improve security, do not store the password in the json, with the risk it leaks , but retrieve it from a global file. Be sure to protect
                        if newConnectionProperties['connection_info']['password']==='':
                            dbKey = '{0}@{1}'.format(newConnectionProperties['connection_info']['user'], newConnectionProperties['connection_info']['database'])
                            newConnectionProperties['connection_info']['password'] = self.databaseParams[dbKey]
                        else:
                            logging.warning("Password is stored in JSON file, please leave it empty")
                        lyr.updateConnectionProperties(oldConnectionProperties, newConnectionProperties, True, validate)
                        replaced  =True
```

{'dataset': 'gis_t.sde_ov.polygon_feature_class', 'workspace_factory': 'SDE', 'connection_info': {'authentication_mode': 'DBMS', 'database': 'gis_t', 'dbclient': 'postgresql', 'db_connection_properties': ‘my.database.local', 'password': '<*********>', 'instance': 'sde:postgresql: my.database.local', 'server': ' my.database.local', 'user': 'sde_ov', 'version': 'sde.DEFAULT'}}

Register datasources with ArcGIS Server 1/2

```
def RegisterDataSources(self,aprx,serverurl):
    gisserver = arcgis.gis.server.Server(url=serverurl , gis=self.gis)
    #servers = gis.admin.servers.list()
    registeredDatasources = gisserver.datastores.list()
    firstmap = aprx.listMaps("*")[0]
    for lyr in firstmap.listLayers("*"):
        if lyr.supports('DATASOURCE'):
            user =lyr.connectionProperties['connection_info']['user']
            database = lyr.connectionProperties['connection_info']['database']
            datasourcename = user + '@' + database
            found=False
            for registeredDatasource in registeredDatasources:
                path = registeredDatasource.properties('path')
                if path.endswith(datasourcename):
                    logging.info("Datasource already registered: " + datasourcename)
                    found = True
            if not found:
                self.RegisterDataSourcefromAprx(lyr,datasourcename,gisserver)


def RegisterDataSourcefromAprx(self, lyr, datasourcename,gisserver):
    path = lyr.dataSource[:lyr.dataSource.rfind('\\')]
    connectionString = self.ConvertSDEtoConnectionString(path,gisserver)
    gisserver.datastores.add_database(datasourcename,connectionString)
    logging.info("Registered new datasource: " + datasourcename)
```

Register datasources with ArcGIS Server 2/2

```python
def ConvertSDEtoConnectionString(self, path, gisserver):
    from arcgis.gis.server.catalog import ServicesDirectory
    from arcgis.gis.server import Uploads

    up = Uploads(url=gisserver.url.replace("admin", "admin/uploads"),
                 gis=gisserver._gis)


    d = ServicesDirectory(url=self.configparams['serverurl'])
    d._con = gisserver._con

    try:
        service = d.get("PublishingTools", 'System')
    except:
        service = d.get("PublishingToolsEx", 'System')
    upload_res = up.upload(path=path)
    if upload_res[0] == True:
        res = service.get_database_connection_string(
            in_conndatatype="UPLOADED_CONNECTION_FILE_ID",
            in_inputdata=upload_res[1]['item']['itemID'])
        up.delete(item_id=upload_res[1]['item']['itemID'])
        return res
```

Create Service draft

```
aprx = arcpy.mp.ArcGISProject(aprxPath)
self.RegisterDataSources(aprx, self.serviceDefinition.serviceDefinition['servers'][self.configparams['server']]['data

 # Provide other service details
serviceName = self.serviceDefinition.getValue('name', self.configparams['server'])
overwrite = self.arcgisTools.ExistsService(self.serviceDefinition.getValue('serverFolder', self.configparams['server'


m = aprx.listMaps("*")[0]
sharing_draft = m.getWebLayerSharingDraft("FEDERATED_SERVER", "MAP_IMAGE", serviceName)
sharing_draft.federatedServerUrl = self.configparams['serverurl']
 # Provide other service details
sharing_draft.summary = self.serviceDefinition.getValue('summary', self.configparams['server'])
sharing_draft.tags = self.serviceDefinition.getValue('tags', self.configparams['server'])
sharing_draft.description = self.serviceDefinition.getValue('description', self.configparams['server'])
sharing_draft.credits = self.serviceDefinition.getValue('credits', self.configparams['server'])
sharing_draft.useLimitations = self.serviceDefinition.getValue('uselimitations', self.configparams['server'])
sharing_draft.portalFolder = self.serviceDefinition.getValue('portalFolder', self.configparams['server'])
sharing_draft.overwriteExistingService =overwrite
sharing_draft.copyDataToServer = False

#create the service draft file
sharing_draft.exportToSDDraft(sddraft)
```

Modify Service Draft and stage Service

```
#remove the unwanted extesions from the draft
# Read the sddraft xml.
doc = DOM.parse(sddraft)
# Find all elements named TypeName. This is where the server object extension (SOE) names are defined,
typeNames = doc.getElementsByTagName('TypeName')
for typeName in typeNames:
    # Get the TypeName we want to remove.
    if typeName.firstChild.data in ['ParcelFabricServer', 'DRServer', 'LRServer','ValidationServer' ]:
        extension = typeName.parentNode

        extensions = extension.parentNode
        extensions.removeChild(extension)
with open(sddraft_mod_xml_file, 'w') as f:
    doc.writexml( f )

os.remove(sddraft)
arcpy.StageService_server(sddraft_mod_xml_file, sd)
```

Publish the service

```
def PublishServiceSd(self):
    sd = os.path.join(self.configparams['pubFolder'], self.serviceDefinition.getValue('name',self.configparams['server']) + ".sd")

    #de serverfolder kan per omgeving een override hebben zodat je OTA op dezelfde server kan hebben
    folder =self.serviceDefinition.getValue('serverFolder',self.configparams['server'])

    try:
        arcpy.UploadServiceDefinition_server(
        in_sd_file=sd,
        in_server=self.configparams['serverurl'],
        in_service_name=self.serviceDefinition.getValue('name',self.configparams['server']),
        in_folder_type='EXISTING',
        in_folder=folder,
        in_startupType='STOPPED')
        logging.debug(arcpy.GetMessages())
```

Move to Portal Folder and share item

```
folder = None
if portalFolder is not None:
    userfolders = self.gis.users.me.folders
    folder = next((x for x in userfolders if x['title'] == portalFolder), None)
    if folder is None:
        folder = self.gis.content.create_folder(portalFolder)
sharing = self.serviceDefinition.serviceDefinition['servers'][self.configparams['server']].get('sharing')
if sharing is not None:
    portalitems = self.arcgisTools.getPortalItemIds( serverFolder ,serviceName, serviceType)
    everyone = sharing.get('esriEveryone') ='true'
    organisation = sharing.get('organisation') ='true'
    groups = sharing.get('groups')
    for itemid in portalitems:
        portalitem =  self.gis.content.get(itemid)
        portalitem.share(everyone=False, org=False, groups=groups, allow_members_to_edit=False)
        if folder is not None:
            portalitem.move(folder)
```

Jenkins

Merkator

Dashboard [Jenkins]

localhost:8080

Jenkins 2 search Joel Hempenius | log out

Jenkins ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- Credentials
- New View

add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		deploy a mapservice	11 days - #36	1 mo 15 days - #17	1 min 3 sec
		deploy a querylayer mapservice	11 days - #5	11 days - #2	49 sec
		deploy TAP	11 days - #2	N/A	4 min 0 sec

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

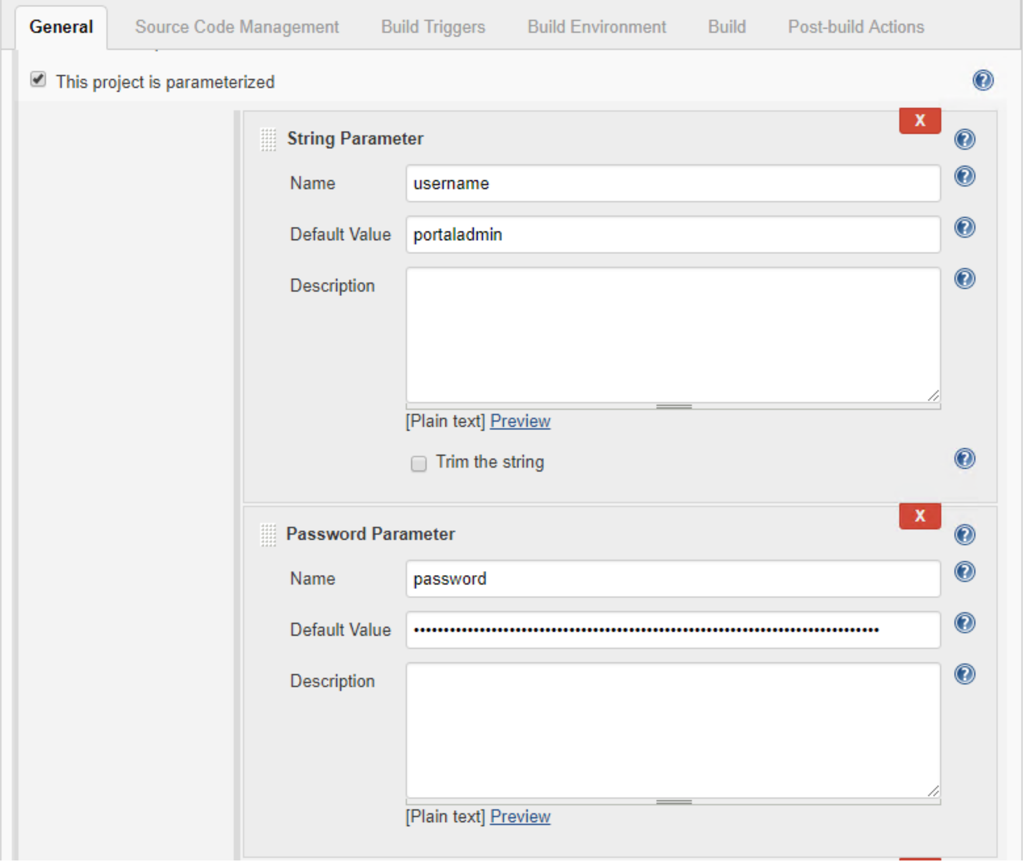
1 Idle
2 Idle

Page generated: Feb 24, 2019 7:13:43 PM CET REST API Jenkins ver. 2.150.1

Jenkins jobs

- Collect build parameters
- Read and checkout SCM (GIT)
- Schedule builds
- Execute build steps
  - Conditional build steps with simple if-then-else blocks
- Collect results
  - Jmeter performance test results
- Email job status

Jenkins jobs 1/4



☑ This project is parameterized

String Parameter

Name: username

Default Value: portaladmin

Description:

☐ Trim the string

Password Parameter

Name: password

Default Value: ••

Description:

Jenkins jobs 2/4

General | **Source Code Management** | Build Triggers | Build Environment | Build | Post-build Actions

Source Code Management

- ○ None
- ◉ Git

Repositories

Repository URL: file:///C:/repos/demo-mapservice/.git

Credentials: - none - | Add

Advanced...

Add Repository

Branches to build

X

Branch Specifier (blank for 'any'): */master

Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

- ○ Subversion

Jenkins jobs 3/4

Merkator

Build

Execute Windows batch command

Command

```
"c:\Program Files\ArcGIS\Pro\bin\Python\scripts\propy.bat" "C:\repos\ci-cd-publishtools-pro\ProPublishTools\InstallMapservice.py" -f %jsonconfig% -s %server% -u %username% -p %password% -r %Datasources% -c %ServiceDefinition% -d %Delete% -h %Stop% -i %Publish% -a %Configure% -z %Start% -j %BUILD_NUMBER% -g %GIT_COMMIT%
```

See the list of available environment variables

Advanced...

Add build step

Jenkins jobs 4/4

Jenkins 2 search Joel Hempenius | log out

Jenkins › deploy a mapservice ›

- Back to Dashboard
- Status
- Changes
- Workspace
- Build with Parameters
- Delete Project
- Configure
- Rename

Build History trend

find

#36	Feb 13, 2019 3:25 PM
#35	Feb 13, 2019 3:23 PM
#34	Feb 13, 2019 3:22 PM
#33	Feb 13, 2019 1:05 PM
#32	Feb 13, 2019 1:04 PM

Project deploy a mapservice

This build requires parameters:

username	portaladmin
password	•••••••••
jsonconfig	TestPublish.aprx.json
server	EDN_T
Datasources	☑
ServiceDefinition	☑
Delete	☐
Stop	☑
Publish	☑
Configure	☑
Start	☑

Build

Jenkins pipelines

- Groovy scripts
  - Define different stages in builds
  - Parallel execution of jobs

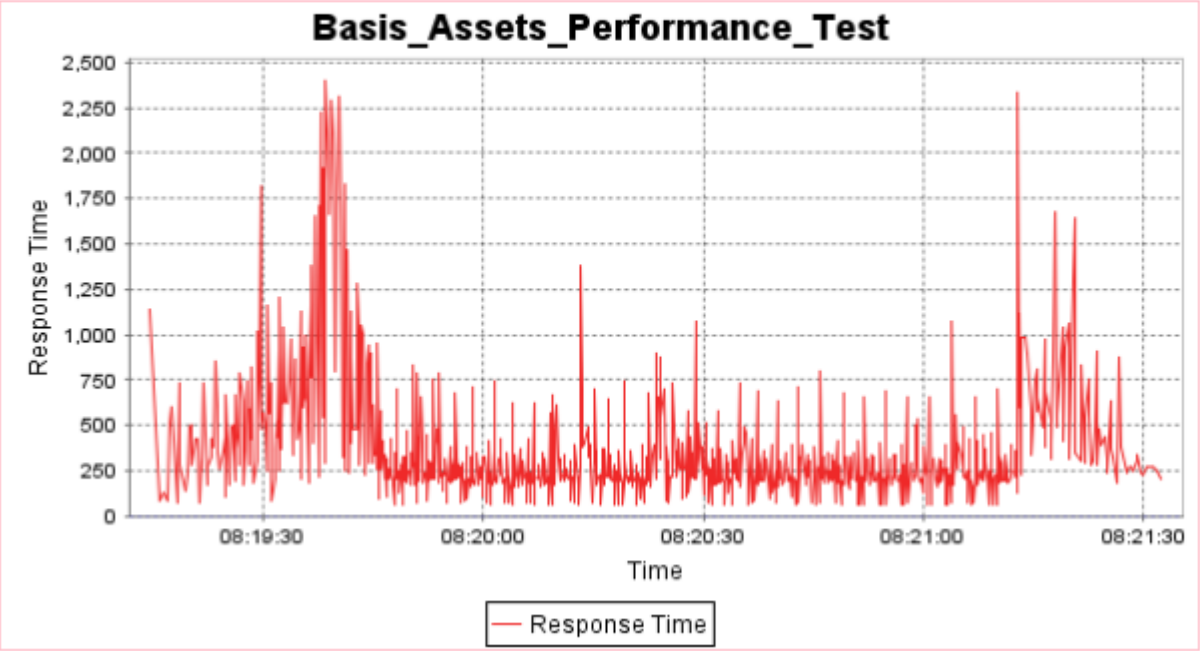
1. Stop mapservices
2. Execute SQL scripts (ALTER TABLE etc)
3. Hit modified schema and tables with schema owner
4. Start mapservices

1. Stop custom print service
2. Execute publish job with only datasources checked, repeat for all different templates
3. Copy updated MXD to server, replacing the print template
4. Start the custom print service

JMeter Performance tests in Jenkins

Merkator

results.jtl

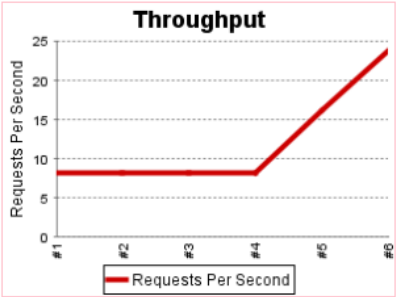


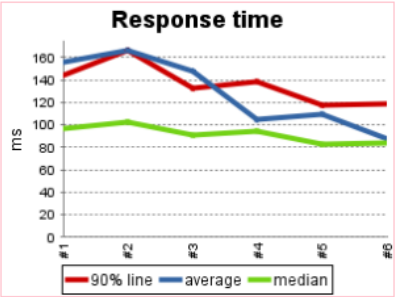
JMeter Performance tests in Jenkins

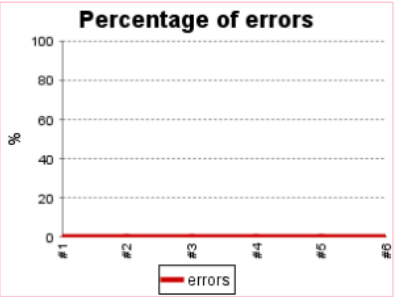
Performance Trend

Last Report
Filter trend data

Test file: results.jtl



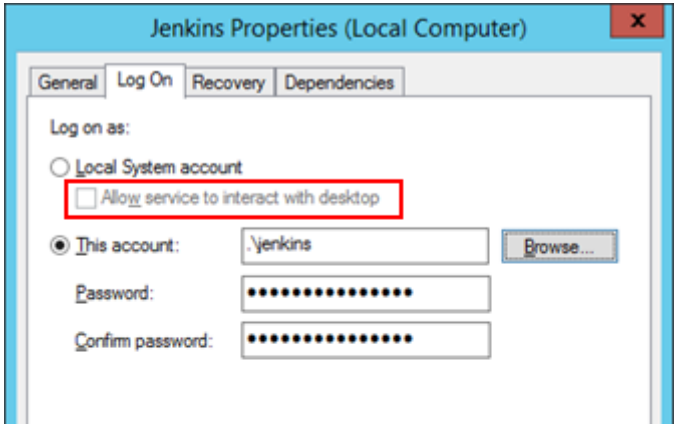




Trend report

Fonts

- Install custom fonts on the Jenkins build server
- Give Jenkins Service Interactive Desktop, otherwise your custom fonts will get lost while publishing



Enable-AllowInteractWithDesktop

Raw

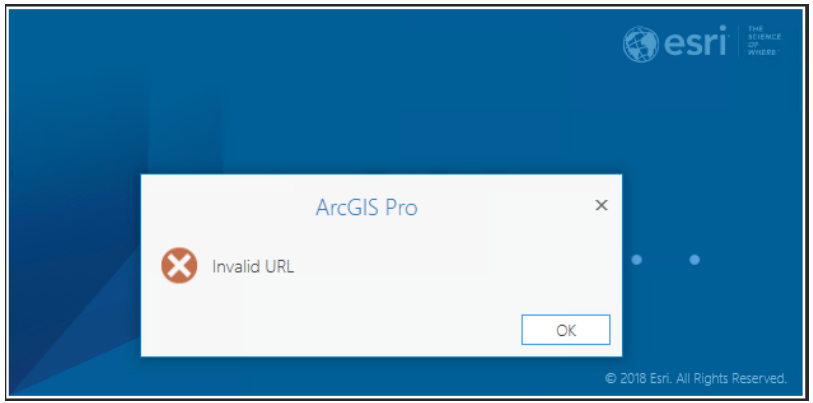
```
$svcName = Get-Service -DisplayName *cruise* | select -Exp Name
$svcKey = Get-Item HKLM:\SYSTEM\CurrentControlSet\Services\$svcName

# Set 9th bit, from http://www.codeproject.com/KB/install/cswindowsservicedesktop.aspx
$newType = $svcKey.GetValue('Type') -bor 0x100
Set-ItemProperty $svcKey.PSPath -Name Type -Value $newType
```

https://lostechies.com/keithdahlby/2011/08/13/allowing-a-windows-service-to-interact-with-desktop-without-localsystem/

Licensing

- Single use or License Manager for ArcMap and Arcgis Pro works with Jenkins
- Named user license from Arcgis Online or Portal not recommended
  - Don’t try to fix this by taking the license offline



Thank you

- Github: https://github.com/PeaceNlove/gispro-jenkins