GlideinWMS Summer 2021 Internship Presentation

Hello!

I am Maia Boyd

I am here today to present the work that I have done this summer.

You can find me at mboyd6@uchicago.edu

Direct Submission Application

Improved the application by making it portable and concise

How it works

- Allows custom scripts to be submitted to an entry from factory
- Each script can be submitted with unique arguments

```
sudo -u gfactory ./gwms_direct_submit.py --entry fermicloud489 --logfile log/test.log --outfile
log/test.out --errfile log/test.err -n 15 legacy/hello.py Hello World How Are You
```

The Specifics

Uses the entry credentials from the factory to modify the entry condor file and generate the environment file

Modifying the Condor File

```
def make_condor(args):
    jobconFile = "/var/lib/gwms-factory/work-dir/entry_" + str(args.ENTRY_NAME) + "/job.condor"
    sub_list = [
        (r"\bLog\s=\s.*", "Log = $ENV(LOGFILE)"),
        (r"\bOutput\s=\s.*", "Output = $ENV(OUTPUTFILE)"),
        (r"\bError\s=\s.*", "Error = $ENV(ERRORFILE)"),
        (r"\bExecutable\s=\s.*", "Executable = $ENV(EXECUTABLE)")
    ]
    tf = tempfile.NamedTemporaryFile("w", delete=False)
    file = open(jobconFile, "r").read() #make a temporary copy of the condor file

    for s in sub_list: #change the temporary file variables
        file = re.sub(s[0], s[1], file)
    tf.writelines(file) #write a new temporary file
    tf.close()

    return tf.name # return the temporary file path

def submit(jobenvFile, jobconFile):
    # Call condor_submit
    p = subprocess.Popen(["/usr/bin/condor_submit", jobconFile], env=jobenvFile, stdout=subprocess.PIPE,
    p.wait()
    return p.stdout.read() #returns the standard output back to caller
```

Generating the Environment File

```
44  def make_environment(args):
45      #logs
46
47      if not os.path.isabs(args.LOGFILE):
48          args.LOGFILE = WORK_DIR + "/" + args.
49      logFile = args.LOGFILE
50        LeRayah Neely, 2 months ago • Add GWMST
51
52      #outputs
53
54      if not os.path.isabs(args.OUTPUTFILE):
55          args.OUTPUTFILE = WORK_DIR + "/" + ar
56      outFile = args.OUTPUTFILE
57
58
59      #errors
60
61      if not os.path.isabs(args.ERRORFILE):
62          args.ERRORFILE = WORK_DIR + "/" + arg
63      errorFile = args.ERRORFILE
64
65
```

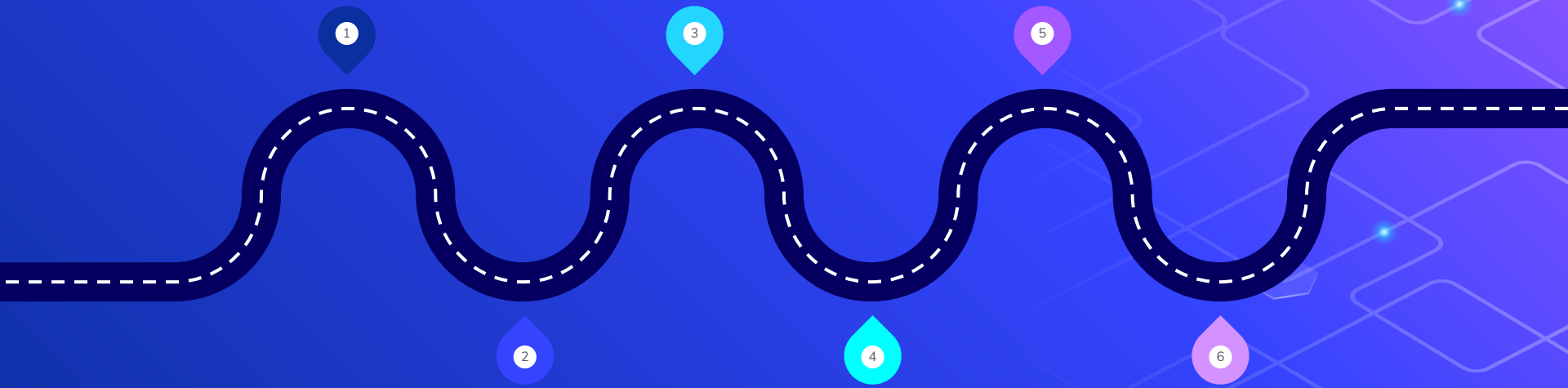
```
66      #executable
67
68      if not os.path.isabs(args.ERRORFILE):
69          args.EXECUTABLE = WORK_DIR + "/" + args.EXECUTABLE
70      execFile = args.EXECUTABLE
71
72
73      # Do some processing
74      print(f"Number of jobs: {args.GLIDEIN_COUNT}")
75      print(args)
76
77      env = generate_submit_environment()
78      env["GLIDEIN_COUNT"] = str(args.GLIDEIN_COUNT)
79      env["LOGFILE"] = logFile
80      env["OUTPUTFILE"] = outFile
81      env["ERRORFILE"] = errorFile
82      env["EXECUTABLE"] = execFile
83      env["ARGUMENTS"] = str(" ".join(exec_args))
84
85      return env
```

Glidein Startup Script

Rewrote the majority of the script in Python

Execution Steps

1. Started by HTCondor and receives command line arguments and initial files
2. Relocates itself to a temporary directory
3. After evaluating arguments, it starts to download additional files
4. The “last script” is executed
5. Collects log information to send back to factory
6. Clean up and exit



```
168 # Cleaup, print out message and exit
169 work_dir_created=0
170 glide_local_tmp_dir_created=0
171
172
173 def glidien_cleanup():
174     # Remove Glidein directories (work_dir, glide_local_tmp_dir)
175     # 1 - exit code
176     # Using GLIDEIN_DEBUG_OPTIONS, start_dir, work_dir_created, work_dir,
177     #   glide_local_tmp_dir_created, glide_local_tmp_dir
178     if not os.path.isabs(start_dir):
179         warnings.warn("Cannot find" + start_dir + "anymore, exiting but without cleanup")
180     else:
181         if GLIDEIN_DEBUG_OPTIONS == nocleanup:
182             warnings.warn("Skipping cleanup, disabled via" + GLIDEIN_DEBUG_OPTIONS)
183         else:
184             if work_dir_created == 1:
185                 os.remove(work_dir)
186             if glide_local_tmp_dir_created == 1:
187                 os.remove(glide_local_tmp_dir)
188
189 # use this for early failures, when we cannot assume we can write to disk at all
190 # too bad we end up with some repeated code, but difficult to do better
191 def early_glidein_failure(error_msg):
192
193     warnings.warn(str(error_msg))
194
195     time.sleep(sleep_time)
196     # wait a bit in case of error, to reduce lost glideins
197     x = datetime.now()
198     glidein_end_time= x.strftime("%S")
199
200     result="    <metric name=\"failure\" ts=\"" + x.strftime("%Y-%m-%dT%H:%M:%S%:z") + "uri=\"local\">WN_RESOURCE</metric> \n<status>ERROR</status> \n<detail>\n\t" +
201
202     final_result= str(construct_xml(result))
203     final_result_simple= str(basexml2simplexml(final_result))
204
205     # have no global section
206     final_result_long= str(simplexml2longxml(final_result_simple, ")", args))
207
208     glidien_cleanup()
209
210     print_tail(1, final_result_simple, final_result_long)
211
212     sys.exit(1)
```

Removes temporary Glidein directories

If glidein fails then run glidein_cleanup, print_tail and exit the program

Thanks!

Any questions?