

**How you
Yes YOU!**

**Can become a JEDI
Developer too**



U.S. AIR FORCE

**Working Principles
GitHub, Git-flow, documentation,
pull requests, code reviews...**








User/Developer Forum



Browser window: forums.jcsda.org

Navigation: all categories ▾ **Categories** Latest Top ⚙️ + New Topic

Category	Topics	Latest
JEDI The Joint Effort for Data assimilation Integration (JEDI) is a unified, next-generation data assimilation system for Earth system prediction.	20	 CRTM Build Issue 2 ■ CRTM 2d
CRTM The Community Radiative Transfer Model (CRTM) is a sensor-based radiative transfer model that supports more than 100 sensors, including those on most meteorological satellites and some from other remote sensing satellites.	7	 Does fv3jedi_convertstate.x run vertical interpolation? 0 ■ JEDI 8d
Uncategorized Topics that don't need a category, or don't fit into any other existing category.	4	 Obtaining quantities like OLR and RSR from CRTM 2 9d
Site Feedback Discussion about this site, its organization, how it works, and how we can improve it.	0	 ObsBias in IODA files 2 ■ JEDI 13d
		 Analysis Increment of JEDI LGETKF 2 ■ JEDI 13d

User/Developer Forum



The image shows two overlapping browser windows from the JCSDA Forums website. The background window displays the main forum page with navigation options like 'all categories', 'Categories', 'Latest', and 'Top'. The foreground window shows a detailed view of the 'JEDI' category, listing various topics with their respective reply counts, view counts, and activity dates.

JEDI

The Joint Effort for Data assimilation Integration (JEDI) is a unified, next-generation data assimilation system for Earth system prediction.

CRTM

The Community Radiative Transfer Model (CRTM) is a sensor-based radiative transfer model that supports more than 100 sensors, including those on most meteorological satellites and some from other remote sensing satellites.

Uncategorized

Topics that don't need a category, or don't fit in any other existing category.

Site Feedback

Discussion about this site, its organization, how it works, and how we can improve it.

Topic	Replies	Views	Activity
Does fv3jedi_convertstate.x run vertical interpolation?	0	9	8d
ObsBias in IODA files	2	20	13d
Analysis Increment of JEDI LGETKF	2	33	13d
JEDI-LETKF/GETLF inflation options	2	64	Apr 6
FV3-JEDI ctest failed with core dumped after turning on "debug"	5	73	Apr 6
3DVar Nan values	0	39	Apr 3

Outline



I) The way of a JEDI

- ◆ Agile project management
- ◆ Collaborative
- ◆ git, GitHub, git-flow

II) Preparing to contribute

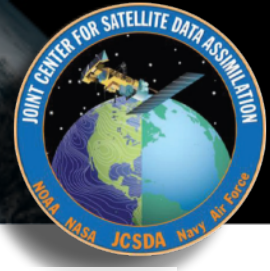
- ◆ Work from a fork
- ◆ Make sure your branch is up to date with develop
- ◆ Make sure your code is adequately tested
- ◆ Make sure your code is adequately documented

III) Contributing code

- ◆ Pull requests
- ◆ Code Reviews



The Way of a JEDI





- ▶ **Collaborative**
 - ◆ **A Joint Center (JCSDA)**
 - **Partners, collaborators, stakeholders, community**
 - ◆ **A Joint Effort (JEDI)**
 - **Distributed team of software developers, with varying objectives and time commitments**

- ▶ **Agile**
 - ◆ **Innovative**
 - ◆ **Flexible (future-proof)**
 - ◆ **Responsive to users and developers**
 - ◆ **Continuous delivery of functional software**

Agile Software Development



► 12 Agile Principles

 <p>Early and continuous delivery of valuable software</p> <p>1</p>	 <p>Welcome changing requirements even late in development</p> <p>2</p>	 <p>Deliver working software frequently</p> <p>3</p>	 <p>Business people and developers working together daily</p> <p>4</p>	 <p>Build projects around motivated individuals and trust them to get the job done</p> <p>5</p>	 <p>The most effective method of conveying information is face-to-face conversation</p> <p>6</p>
 <p>Working software is the primary measure of progress</p> <p>7</p>	 <p>Sustainable development: maintain a constant pace indefinitely</p> <p>8</p>	 <p>Continuous attention to technical excellence</p> <p>9</p>	 <p>Simplicity: maximize the amount of work not done</p> <p>10</p>	 <p>Teams self-organize</p> <p>11</p>	 <p>Teams regularly reflect and adjust behaviour</p> <p>12</p>

Agile Software Development



► 12 Agile Principles



Early and continuous delivery of valuable software

1



Welcome changing requirements even late in development

2




Deliver working software frequently

3




Business people and developers working together daily

4



Build projects around motivated individuals and trust them to get the job done

5




The most effective method of conveying information is face-to-face conversation

6




Working software is the primary measure of progress

7



Sustainable development: maintain a constant pace indefinitely

8



Continuous attention to technical excellence

9




Simplicity: maximize the amount of work not done

10



Teams self-organize

11



Teams regularly reflect and adjust behaviour

12

Agile Software Development



► 12 Agile Principles




Early and continuous delivery of valuable software

1




Welcome changing requirements even late in development

2




Deliver working software frequently

3




Business people and developers working together daily

4




Build projects around motivated individuals and trust them to get the job done

5




The most effective method of conveying information is face-to-face conversation

6




Working software is the primary measure of progress

7



Sustainable development: maintain a constant pace indefinitely

8




Continuous attention to technical excellence

9



Simplicity: maximize the amount of work not done

10



Teams self-organize

11



Teams regularly reflect and adjust behaviour

12

Agile Software Development



► 12 Agile Principles



Early and continuous delivery of valuable software

1




Welcome changing requirements even late in development

2




Deliver working software frequently

3




Business people and developers working together daily

4




Build projects around motivated individuals and trust them to get the job done

5




The most effective method of conveying information is face-to-face conversation

6




Working software is the primary measure of progress

7



Sustainable development: maintain a constant pace indefinitely

8




Continuous attention to technical excellence

9




Simplicity: maximize the amount of work not done

10



Teams self-organize

11



Teams regularly reflect and adjust behaviour

12

Agile Software Development



► 12 Agile Principles

 <p>Early and continuous delivery of valuable software</p> <p>1</p>	 <p>Welcome changing requirements even late in development</p> <p>2</p>	 <p>Deliver working software frequently</p> <p>3</p>	 <p>Business people and developers working together daily</p> <p>4</p>	 <p>Build projects around motivated individuals and trust them to get the job done</p> <p>5</p>	 <p>The most effective method of conveying information is face-to-face conversation</p> <p>6</p>
 <p>Working software is the primary measure of progress</p> <p>7</p>	 <p>Sustainable development: maintain a constant pace indefinitely</p> <p>8</p>	 <p>Continuous attention to technical excellence</p> <p>9</p>	 <p>Simplicity: maximize the amount of work not done</p> <p>10</p>	 <p>Teams self-organize</p> <p>11</p>	 <p>Teams regularly reflect and adjust behaviour</p> <p>12</p>

Agile Software Development



► 12 Agile Principles

 <p>Early and continuous delivery of valuable software</p> <p>1</p>	 <p>Welcome changing requirements even late in development</p> <p>2</p>	 <p>Deliver working software frequently</p> <p>3</p>	 <p>Business people and developers working together daily</p> <p>4</p>	 <p>Build projects around motivated individuals and trust them to get the job done</p> <p>5</p>	 <p>The most effective method of conveying information is face-to-face conversation</p> <p>6</p>
 <p>Working software is the primary measure of progress</p> <p>7</p>	 <p>Sustainable development: maintain a constant pace indefinitely</p> <p>8</p>	 <p>Continuous attention to technical excellence</p> <p>9</p>	 <p>Simplicity: maximize the amount of work not done</p> <p>10</p>	 <p>Teams self-organize</p> <p>11</p>	 <p>Teams regularly reflect and adjust behaviour</p> <p>12</p>

Agile Software Development



► 12 Agile Principles

 <p>Early and continuous delivery of valuable software</p> <p>1</p>	 <p>Welcome changing requirements even late in development</p> <p>2</p>	 <p>Deliver working software frequently</p> <p>3</p>	 <p>Business people and developers working together daily</p> <p>4</p>	 <p>Build projects around motivated individuals and trust them to get the job done</p> <p>5</p>	 <p>The most effective method of conveying information is face-to-face conversation</p> <p>6</p>
 <p>Working software is the primary measure of progress</p> <p>7</p>	 <p>Sustainable development: maintain a constant pace indefinitely</p> <p>8</p>	 <p>Continuous attention to technical excellence</p> <p>9</p>	 <p>Simplicity: maximize the amount of work not done</p> <p>10</p>	 <p>Teams self-organize</p> <p>11</p>	 <p>Teams regularly reflect and adjust behaviour</p> <p>12</p>

Agile Software Development



► 12 Agile Principles

 <p>Early and continuous delivery of valuable software</p> <p>1</p>	 <p>Welcome changing requirements even late in development</p> <p>2</p>	 <p>Deliver working software frequently</p> <p>3</p>	 <p>Business people and developers working together daily</p> <p>4</p>	 <p>Build projects around motivated individuals and trust them to get the job done</p> <p>5</p>	 <p>The most effective method of conveying information is face-to-face conversation</p> <p>6</p>
 <p>Working software is the primary measure of progress</p> <p>7</p>	 <p>Sustainable development: maintain a constant pace indefinitely</p> <p>8</p>	 <p>Continuous attention to technical excellence</p> <p>9</p>	 <p>Simplicity: maximize the amount of work not done</p> <p>10</p>	 <p>Teams self-organize</p> <p>11</p>	 <p>Teams regularly reflect and adjust behaviour</p> <p>12</p>

Agile Tools



▸ **git/GitHub**

- ◆ **Version control and Release distribution**
- ◆ **Pull requests, Code reviews**
- ◆ **Coordination of distributed community of developers**

▸ **Git-Flow**

- ◆ **Innovation**
- ◆ **Continuous Delivery**

▸ **ZenHub**

- ◆ **Agile project management**
- ◆ **Issue tracking, enhanced code review**

▸ **Forums: <https://forums.jcsda.org>**

- ◆ **User support, stakeholder feedback**

git/GitHub



Browser: JCSDA/oops: Object Oriented | x +

github.com/JCSDA/oops

Search or jump to... Pull requests Issues Marketplace Explore

JCSDA / oops

Watch 21 Star 1 Fork 1

Code Pull requests Actions ZenHub Security Insights Settings

master 2 branches 3 tags

Go to file Add file Code

About

Object Oriented Prediction System

Readme Apache-2.0 License

Releases 3

1.1.0 Latest 13 days ago

+ 2 releases

Packages

No packages published

File/Folder	Commit Message	Time Ago
.github/ISSUE_TEMPLATE	add issue templates (#1075)	4 months ago
CI	update ci containers (#1232)	23 days ago
cmake	update compare tests in I95 and qg (#1178)	2 months ago
docs	update doxyfile (#10)	8 months ago
ewok	Feature/hofx4d ewok (#1197)	2 months ago
I95	Bugfix/fix comparefloat (#1231)	21 days ago
qg	comment out saddlepoint test (#1239)	21 days ago
src	Add a call to ObsSpace::save() from ObsTestsFixture::rese...	20 days ago
tools	remove output file before running test (#1105)	3 months ago

mmiesch Merge pull request #1241 from JCSDA-internal/rel... 0a07af4 13 days ago 1,128 commits

Git-branching-m....pdf Show All

git/GitHub



The screenshot shows a web browser displaying the GitHub repository page for 'JCSDA/oops'. The page is in dark mode. At the top, there's a search bar and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below that, the repository name 'JCSDA / oops' is shown with a menu icon. A navigation bar includes 'Code', 'Pull requests', 'Actions', 'ZenHub', 'Security', and 'Insights'. The main content area shows a merge pull request by 'mmiesch' from 'JCSDA-internal/rel...' to 'master', dated '13 days ago'. Below the pull request is a file tree with the following items:

File/Folder	Description
.github/ISSUE_TEMPLATE	add issue templates (#1075)
CI	update ci containers (#1232)
cmake	update compare tests in I95 and qg (#1178)
docs	update doxyfile (#10)
ewok	Feature/hofx4d ewok (#1197)
I95	Bugfix/fix comparefloat (#1231)
qg	comment out saddlepoint test (#1239)
src	Add a call to ObsSpace::save() from ObsTestsFixture::rese...
tools	remove output file before running test (#1105)

At the bottom of the page, there's a 'No packages published' message and a 'Show All' button.

***git - command line tool
(version control)***

***GitHub - Web-based
repository management
(branches, releases)***

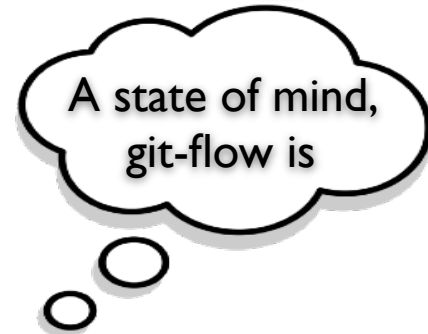
***Changes to develop, master
branches handled via
pull requests***

Git-Flow



Git Flow is:

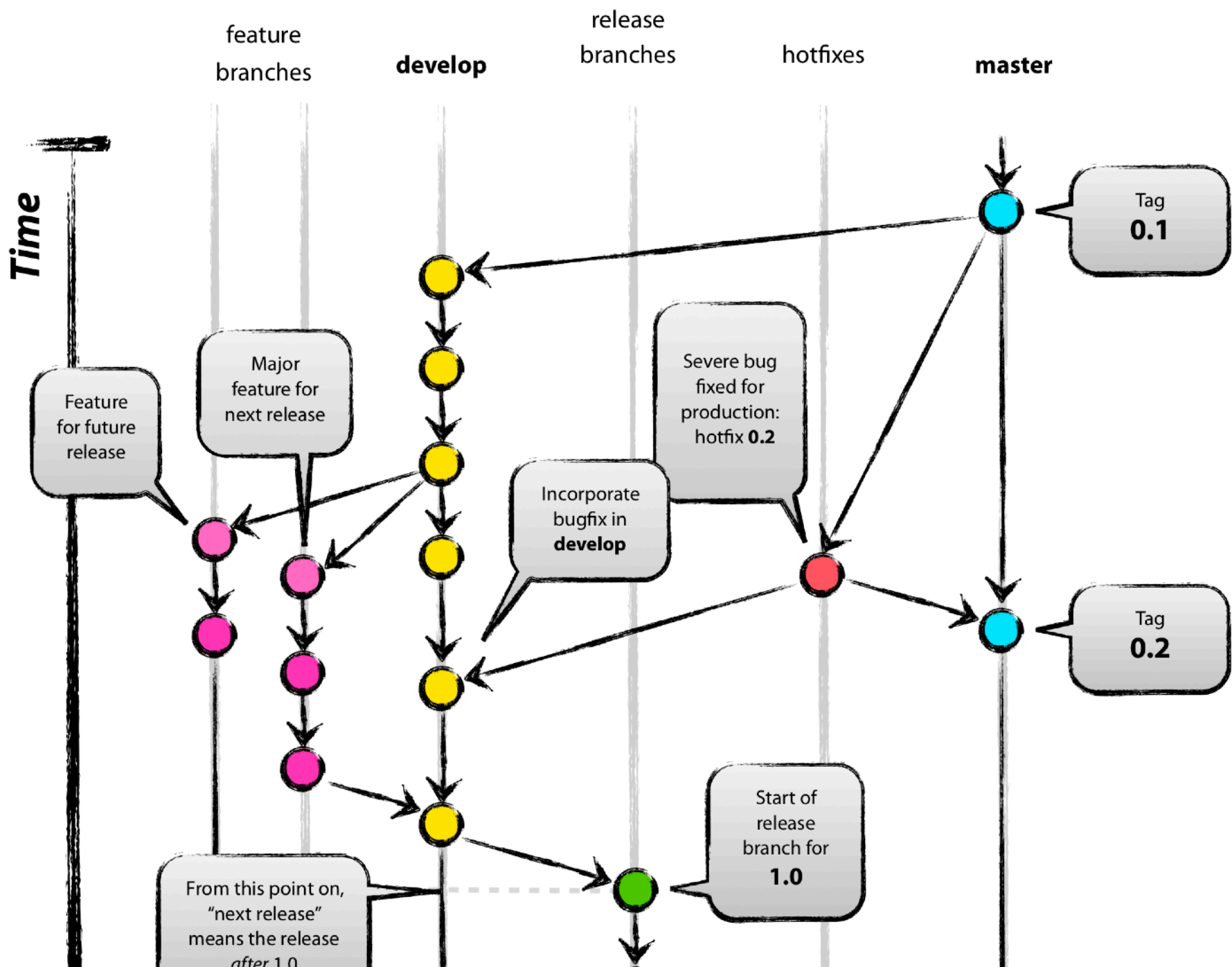
- ▶ **A Philosophy**
 - ◆ **Optimal for Agile Software Development**
 - **Innovation**
 - **Continuous Delivery**
- ▶ **A Working Principle**
 - ◆ **Enforcement of branch naming conventions**
- ▶ **An Application (*extension to git*)**
 - ◆ **Already installed Singularity Container**



Vincent
Driessen
(2010)

Git-flow manifesto

<http://nvie.com/posts/a-successful-git-branching-model/>



Life Cycle of a Feature branch



- 1) Enable git flow for the repo
 - **git flow init -d**
- 2) Start the feature branch
 - **git flow feature start newstuff**
 - Creates a new branch called feature/newstuff that branches off of develop
- 3) Push it to GitHub for the first time
 - Make changes and commit them locally
 - **git flow feature publish newstuff**
- 4) Additional (normal) commits and pushes as needed
 - **git commit -a**
 - **git push**
- 5) Bring it up to date with develop (to minimize big changes on the ensuing pull request)
 - **git checkout develop**
 - **git pull origin develop**
 - **git checkout feature/newstuff**
 - **git merge develop**
- 6) Finish the feature branch (**don't use git flow feature finish**)
 - Do a pull request on GitHub from feature/newstuff to develop
 - When successfully merged the remote branch will be deleted
 - **git remote update -p**
 - **git branch -D feature/newstuff**

Life Cycle of a Feature branch



- 1) Enable git flow for the repo
 - **git flow init -d**
- 2) Start the feature branch
 - **git flow feature start newstuff**
 - Creates a new branch called feature/newstuff that branches off of develop
- 3) Push it to GitHub for the first time
 - Make changes and commit them locally
 - **git flow feature publish newstuff**
- 4) Additional (normal) commits and pushes as needed
 - **git commit -a**
 - **git push**
- 5) Bring it up to date with develop (to minimize big changes)
 - **git checkout develop**
 - **git pull origin develop**
 - **git checkout feature/newstuff**
 - **git merge develop**
- 6) Finish the feature branch (**don't use git flow feature finish**)
 - Do a pull request on GitHub from feature/newstuff to develop
 - When successfully merged the remote branch will be updated
 - **git remote update -p**
 - **git branch -D feature/newstuff**

***What if I can't install
git-flow?***

***Just be sure to use the
proper naming and
branching conventions***

**feature/mybranch
release/mybranch
bugfix/mybranch
hotfix/mybranch**

The Git-Flow Manifesto: Takaways



- ▶ **master is for releases only**
- ▶ **develop**
 - **Not ready for public consumption but compiles and passes all tests**
- ▶ **Feature branches**
 - **Where most development happens**
 - **Branch off of develop**
 - **Merge into develop**
- ▶ **Release branches**
 - **Branch off of develop**
 - **Merge into master and develop**
- ▶ **Hotfix**
 - **Branch off of master**
 - **Merge into master and develop**
- ▶ **Bugfix**
 - **Branch off of develop**
 - **Merge into develop**

Feature branches should be focused and short, with a specific goal

They should exist for days or weeks, not months



I) The way of a JEDI

- ◆ Agile project management
- ◆ Collaborative
- ◆ git, GitHub, git-flow

II) Preparing to contribute

- ◆ Work from a fork
- ◆ Make sure your branch is up to date with develop
- ◆ Make sure your code is adequately tested
- ◆ Make sure your code is adequately documented

III) Contributing code

- ◆ Pull requests
- ◆ Code Reviews



Part II: Preparing to contribute



The screenshot shows the GitHub interface for the repository `JCSDA/ufo`. The page includes a search bar, navigation tabs for Pulls, Issues, Marketplace, and Explore, and a header with repository statistics: 25 watchers, 0 stars, and 2 forks. The main content area displays a list of pull requests, with the most recent one being a merge of the 'master' branch into 'develop' by user `ytremolet`, completed 16 days ago with 2,146 views. Below this, a list of folders and files is shown, each associated with a pull request number and its completion date. The right sidebar contains an 'About' section with the repository name 'Unified Forward Operator', a 'Readme' link, and the license 'Apache-2.0 License'. It also features a 'Releases' section with one release, version `1.0.0`, marked as 'Latest' and released 16 days ago. At the bottom, a 'Packages' section indicates that no packages have been published.

File/Folder	Description	Created
CI	clean up; change crtm tag; try cloning depth 1 (#25)	17 days ago
cmake	removed underflow flag from compiler options	2 years ago
docs	add doxygen build (#19)	19 days ago
src	Feature/gnssro match gsi superrefraction (#12)	21 days ago
test	make sure output files don't write into test data dirs ...	17 days ago
tools	Update scripts generating observation operators a...	28 days ago
.gitattributes	Feature/sonde consistency checks (#834)	6 months ago

Part II: Preparing to contribute



The screenshot shows the GitHub interface for the repository `JCSDA / ufo`. The repository has 25 watchers, 0 stars, and 2 forks. The 'Fork' button is highlighted with a blue circle. Below the repository name, there are navigation tabs for Code, Pull requests, Actions, Security, and Insights. The main content area shows a list of commits, with the most recent one by `ytremolet` merging 'master' into 'develop' 16 days ago. The commit list includes folders like `CI`, `cmake`, `docs`, `src`, `test`, and `tools`, as well as a `.gitattributes` file. On the right side, there is an 'About' section with the repository name 'Unified Forward Operator', a 'Readme' link, and an 'Apache-2.0 License'. Below that, there is a 'Releases' section showing version `1.0.0` as the latest release, published 16 days ago. At the bottom, there is a 'Packages' section indicating that no packages have been published.

Commit	Description	Time
<code>ytremolet</code>	Merge branch 'master' into develop	16 days ago
<code>CI</code>	clean up; change crtm tag; try cloning depth 1 (#25)	17 days ago
<code>cmake</code>	removed underflow flag from compiler options	2 years ago
<code>docs</code>	add doxygen build (#19)	19 days ago
<code>src</code>	Feature/gnssro match gsi superrefraction (#12)	21 days ago
<code>test</code>	make sure output files don't write into test data dirs ...	17 days ago
<code>tools</code>	Update scripts generating observation operators a...	28 days ago
<code>.gitattributes</code>	Feature/sonde consistency checks (#834)	6 months ago

Part II: Preparing to contribute



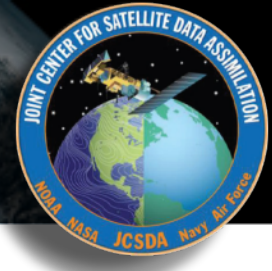
The screenshot shows the GitHub interface for the repository 'JCSDA / ufo'. The 'Fork' button is circled in blue. Below the repository name, there are buttons for 'Code', 'Pull requests', 'Actions', 'Security', and 'Insights'. The repository is currently on the 'master' branch. A commit history table is visible, showing a merge of 'master' into 'develop' and several other commits with their descriptions and issue numbers.

Commit	Description	Issue
ytremolet	Merge branch 'master' into develop	
CI	clean up; change crtm tag; try cloning depth 1	#25
cmake	removed underflow flag from compiler options	
docs	add doxygen build	#19
src	Feature/gnssro match gsi superrefraction	#12
test	make sure output files don't write into test data dirs ...	
tools	Update scripts generating observation operators a...	
.gitattributes	Feature/sonde consistency checks	#834

First - fork the repository or repositories you would like to work with

This may be a personal or an institutional fork

Create a feature branch



Set up JCSDA as the develop branch

```
git clone https://github.com/<myaccount>/ufo.git
cd ufo
git remote add upstream https://github.com/JCSDA/ufo.git
git fetch --tags upstream
git checkout --track upstream/develop
```

Create feature branch from JCSDA develop

```
git checkout -b feature/<mybranch> develop
```

Implement code changes



Edit the code in the feature branch, commit changes, and push it to your fork

```
git add *  
git status  
git commit  
git push origin --set-upstream feature/<mybranch>
```

Make sure you're committing the files you intend to commit

Continue to make changes, commit them, test them, and push to your fork. Periodically synchronize with JCSDA develop and resolve any merge conflicts that may arise

```
git checkout develop  
git pull upstream develop  
git checkout feature/<mybranch>  
git merge develop
```

Add Tests and Documentation



Be sure to add **tests** that execute the code you added or modified
(For instructions, see Maryam's lecture)

If you do not, then your code will not pass our CI (CodeCov) testing and it will not be merged

Also add **documentation** explaining the purpose of the code, what it does, how to use it, when to use it, scientific and/or mathematical background, and known limitations or bugs

▸ **Doxygen**

✦ **Low-level descriptions of functions, classes, subroutines, etc, embedded directly in the code**

▸ **Sphinx: <http://jedi-docs.jcsda.org>**

✦ **Repository: <https://github.com/JCSDA/jedi-docs.git>**

✦ **High-level documentation (context, use cases, theory...)**

Documenting Fortran Source Code



```
!! _____
!!> \brief Example function
!!
!! \details myfunction() takes a and b as arguments and miraculously creates c.
!! I could add many more details here if I chose to do so. I can even make a list:
!! * item 1
!! * item 2
!! * item 3
!!
!! \date A long, long, time ago: Created by L. Skywalker (JCSDA)
!!
!! \warning This isn't a real function!
!!
subroutine myfunction(a, b, c)
  integer, intent(in)      :: a !< this is one input parameter
  integer, intent(in)      :: b !< this is another
  real(kind=kind_rea), intent(out) :: c !< and this is the output
  [...]
```



Note
**Doxygen has known problems
with object-oriented Fortran and
Fortran/C++ bindings**

Documenting C++ Source Code



```
// -----  
/!* \brief Example function  
*  
* \details **myfunction()** takes a and b as arguments and miraculously creates c.  
* I could add many more details here if I chose to do so. I can even make a list:  
* * item 1  
* * item 2  
* * item 3  
*  
* \param[in] a this is one input parameter  
* \param[in] b this is another  
* \param[out] c and this is the output  
*  
* \date A long, long, time ago: Created by L. Skywalker (JCSDA)  
*  
* \warning This isn't a real function!  
*  
*/  
void myfunction(int& a, int& b, double& c) {  
    [...]
```



Useful Doxygen Commands



- ▶ `\brief`
- ▶ `\details`
- ▶ `\param`
- ▶ `\return`
- ▶ `\author`
- ▶ `\date`
- ▶ `\note`
- ▶ `\attention`
- ▶ `\warning`
- ▶ `\bug`
- ▶ `\class <name> [<header-file>]`
- ▶ `\mainpage`
- ▶ `\f$... \f$` (**inline formula**)
- ▶ `\f[... \f]` (**formula block**)
- ▶ `\em` (**or * ... ***)
- ▶ `\sa` (**see also**)
- ▶ `\typedef`
- ▶ `\todo`
- ▶ `\version`
- ▶ `\namespace`
- ▶ `...` (**url**)
- ▶ `\image`
- ▶ `\var`
- ▶ `\throws` (**exception description**)

Many more described here:

<https://www.stack.nl/~dimitri/doxygen/manual/commands.html>

Sample output: “man page”



◆ testStateInterpolation()

template<typename MODEL >

```
void test::testStateInterpolation ( )
```

Interpolation test.

testStateInterpolation() tests the interpolation for a given model. The conceptual steps are as follows:

1. Initialize the JEDI **State** object based on idealized analytic formulae
2. Interpolate the **State** variables onto selected "observation" locations using the `getValues()` method of the **State** object. The result is placed in a JEDI **GeoVaLs** object
3. Compute the correct solution by applying the analytic formulae directly at the observation locations.
4. Assess the accuracy of the interpolation by comparing the interpolated values from Step 2 with the exact values from Step 3

The interpolated state values are compared to the analytic solution for a series of **locations** which includes values optionally specified by the user in the "StateTest" section of the config file and a randomly-generated list of **Nrandom** random locations. Nrandom is also specified by the user in the "StateTest" section of the config file, as is the (nondimensional) tolerance level (**intp_tol**) to be used for the tests.

This is an equation:

$$\zeta = \left(\frac{x - x_0}{\lambda} \right)^{2/3}$$

Relevant parameters in the ****State*** section of the config file include

- **norm-gen** Normalization test for the generated **State**
- **interp_tolerance** tolerance for the interpolation test

Date

April, 2018: M. Miesch (JCSDA) adapted a preliminary version in the feature/interp branch

Warning

Since this model compares the interpolated state values to an exact analytic solution, it requires that the "analytic_init" option be implemented in the model and selected in the "State.StateGenerate" section of the config file.

Corresponding code



```
// -----  
/!* \brief Interpolation test  
*  
* \details **testStateInterpolation()** tests the interpolation for a given  
* model. The conceptual steps are as follows:  
* 1. Initialize the JEDI State object based on idealized analytic formulae  
* 2. Interpolate the State variables onto selected "observation" locations  
* using the getValues() method of the State object. The result is  
* placed in a JEDI GeoVaLs object  
* 3. Compute the correct solution by applying the analytic formulae directly  
* at the observation locations.  
* 4. Assess the accuracy of the interpolation by comparing the interpolated  
* values from Step 2 with the exact values from Step 3  
*  
* The interpolated state values are compared to the analytic solution for  
* a series of **locations** which includes values optionally specified by the  
* user in the "StateTest" section of the config file in addition to a  
* randomly-generated list of **Nrandom** random locations. Nrandom is also  
* specified by the user in the "StateTest" section of the config file, as is the  
* (nondimensional) tolerance level (**interp_tolerance**) to be used for the tests.  
[...]
```

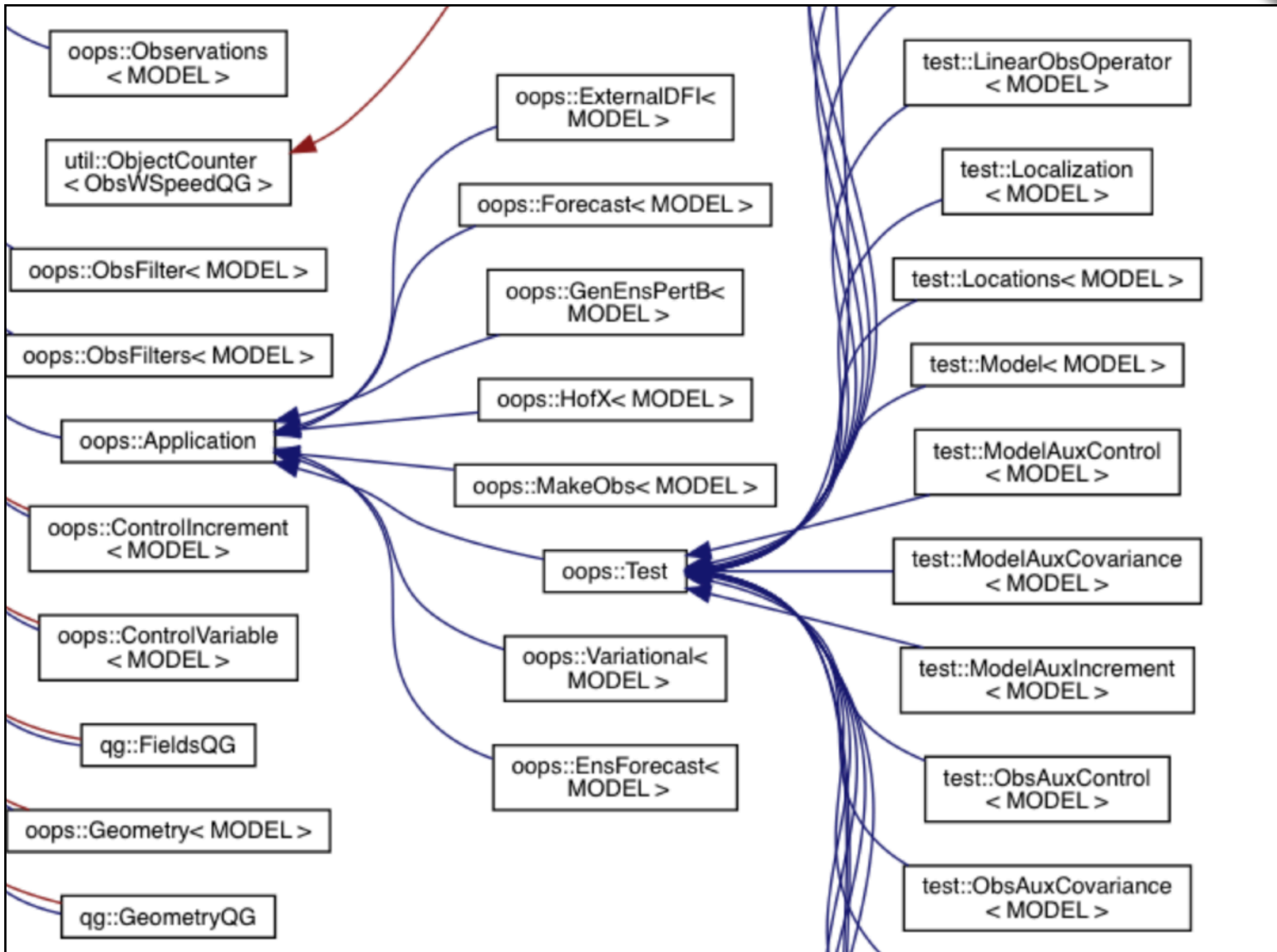
Corresponding code (cont.)



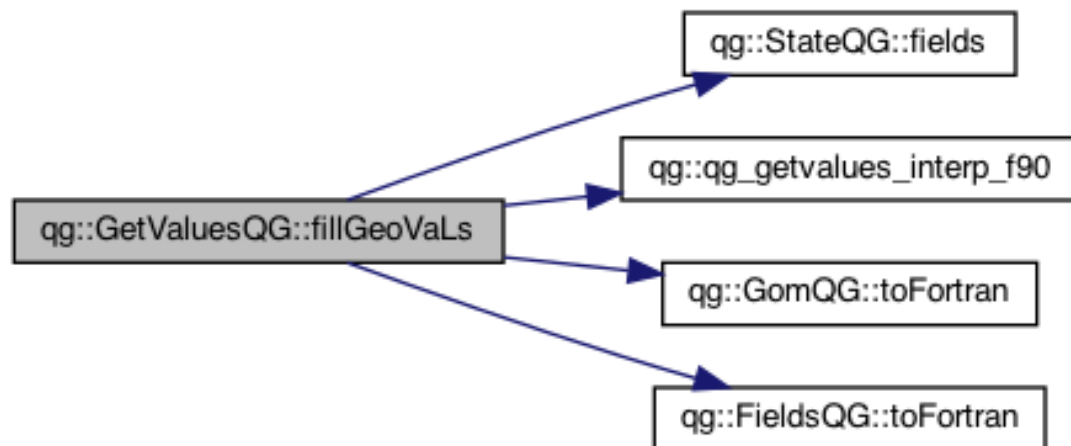
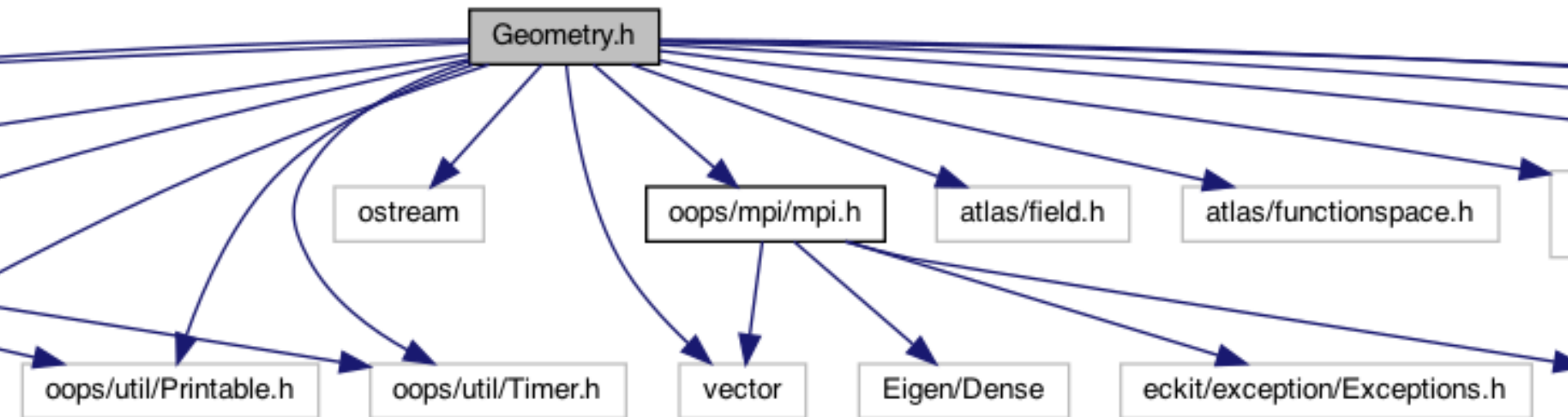
[...]

```
*  
  
* This is an equation:  
*  $\zeta = \left(\frac{x-x_0}{\lambda}\right)^{2/3}$   $\zeta$   
*  
* Relevant parameters in the **State* section of the config file include  
*  
* * **norm-gen** Normalization test for the generated State  
* * **interp_tolerance** tolerance for the interpolation test  
*  
* \date April, 2018: M. Miesch (JCSDA) adapted a preliminary version in the  
* feature/interp branch  
*  
* \warning Since this model compares the interpolated state values to an exact analytic  
* solution, it requires that the "analytic_init" option be implemented in the model and  
* selected in the "State.StateGenerate" section of the config file.  
*/
```

Sample output: class hierarchy

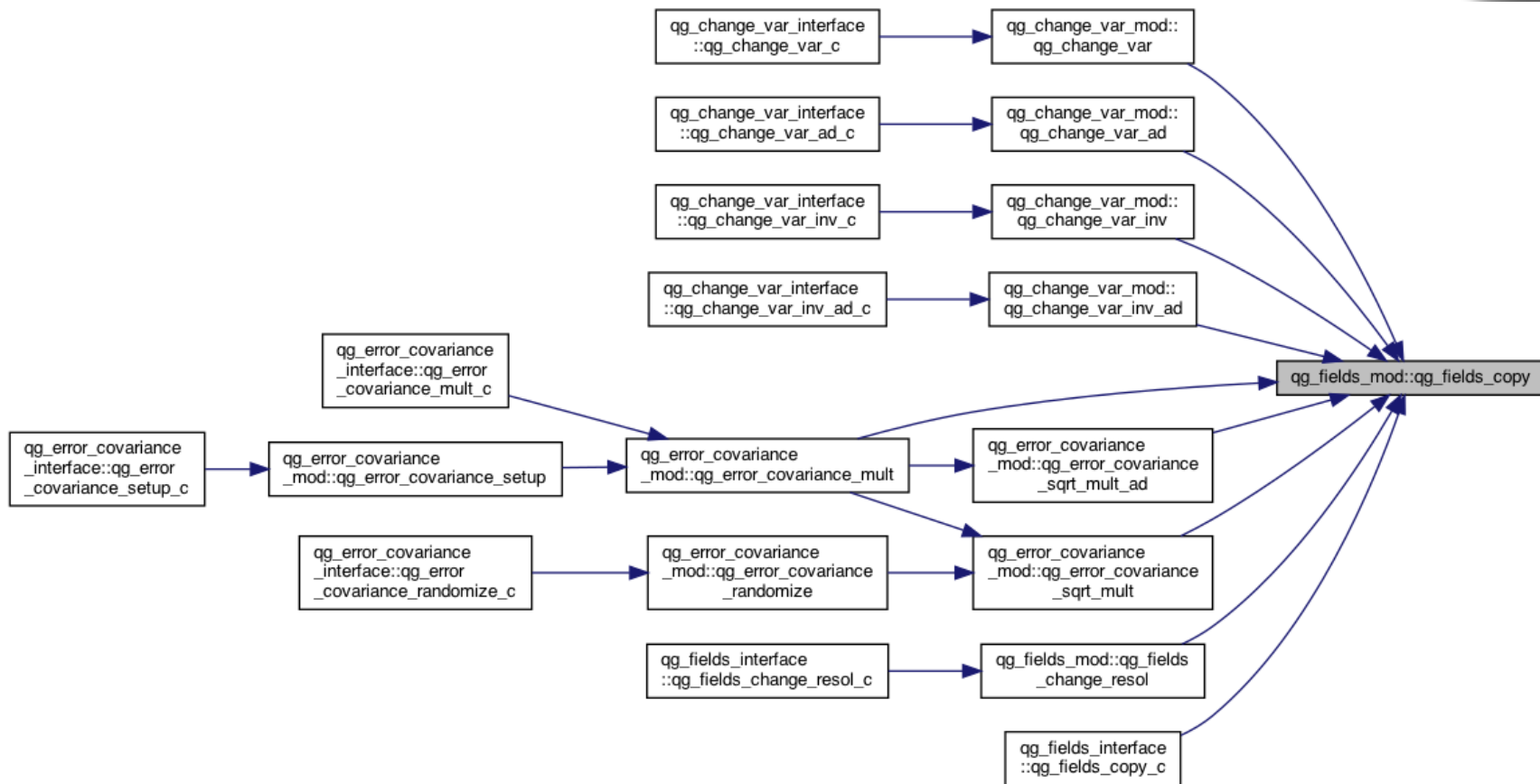


Sample output: include, call graphs



Clickable boxes!

Sample output: caller graphs



Note that these traces end in `_c` (this is a Fortran routine)
Doxygen has trouble with C++ / Fortran binding
Look for corresponding `_f90` routine to follow further

Doxygen in JEDI



After you have added doxygen documentation to the source code, you can generate html doxygen output for a particular repository by enabling the documentation with ecbuild.

Be sure you have **doxygen** and **graphviz** installed (can install with **homebrew**, **apt**, **yum**, etc)

```
ecbuild -DENABLE_OOPS_DOC=ON ../fv3-bundle  
make -j4
```

You can find the results in the **<build>/<repo>/docs/html** directory

Doxygen documentation for JEDI components is available on the academy and JEDI documentation web sites

JEDI User/Developer Manual



JEDI Documentation
latest

Search docs

- Overview
- Working Principles
- Learning JEDI
- Using JEDI
- Inside JEDI
- Frequently Asked Questions (FAQ)
- References

Read the Docs v: latest

Docs » JEDI Documentation [Edit on GitHub](#)

JEDI Documentation

Welcome to the Joint Effort for Data assimilation Integration (JEDI)!

JEDI is a unified and versatile **data assimilation (DA)** system for Earth System Prediction. The JEDI software package can be run on a variety of platforms from laptops to supercomputers, for a variety of purposes, from teaching and learning DA fundamentals to the development and validation of new DA algorithms and observational operators, to leading-edge atmospheric and oceanic research, to operational weather forecasting. It is designed to readily accommodate new atmospheric and oceanic models and new observation systems.

JEDI is developed and distributed by the [Joint Center for Satellite Data Assimilation](#), a [multi-agency](#) research center hosted by [the University Corporation for Atmospheric Research \(UCAR\)](#). JCSDA is dedicated to improving and accelerating the quantitative use of research and operational satellite data in weather, ocean, climate and environmental analysis and prediction systems.

JEDI is a community effort and external contributions are welcome. This document serves as both a user manual and a developers' guide.

If you have questions about the JEDI project, JEDI usage, JEDI components,

JEDI User/Developer Manual



The screenshot shows a web browser displaying the JEDI Documentation website. The browser's address bar shows the URL: `https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com`. The page title is "JEDI Documentation" and the version is "latest".

The navigation menu on the left includes the following items:

- Overview
- Working Principles
- Learning JEDI
- Using JEDI
- Inside JEDI
- Frequently Asked Questions (FAQ)
- References

The main content area displays the "JEDI Documentation" header and a search bar. Below the header, there is a section titled "JEDI is a unified..." and another section titled "JEDI is developed...".

The "JEDI is developed..." section contains the following text:

JEDI is developed by the [Joint Center for Satellite Data Assimilation](#), a multi-agency effort led by the [Air Force Research Laboratory](#) and the [Naval Research Laboratory](#). JEDI is designed to improve and accelerate the assimilation of satellite data in weather prediction systems.

The "JEDI is developed..." section also includes a list of JEDI components:

- JOOPS
- SABER
- BUMP
- IODA
- UFO
- FV3-JEDI
- Configuring JEDI

The "JEDI is a unified..." section contains the following text:

JEDI is a unified architecture for weather and climate Prediction. The JEDI architecture is designed to run from laptops to supercomputers, providing a learning DA fundamental algorithms and observations from oceanic research, and to accommodate new satellite systems.

The "JEDI is a unified..." section also includes the following text:

JEDI is a community effort. The JEDI documentation serves as both a user manual and a developers' guide.

If you have questions about the JEDI project, JEDI usage, JEDI components,

The screenshot shows a web browser displaying the JEDI Documentation website, specifically the "BUMP" section. The browser's address bar shows the URL: `https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com`. The page title is "Goals and code organization" and the version is "latest".

The navigation menu on the left includes the following items:

- Overview
- Working Principles
- Learning JEDI
- Using JEDI
- Inside JEDI
 - JEDI Components
 - JOOPS
 - SABER
 - BUMP
 - IODA
 - UFO
 - FV3-JEDI
 - Configuring JEDI

The main content area displays the "BUMP" section, which includes the following text:

Three categories of background error covariance models are currently implemented in BUMP:

- The ensemble covariance model is built as a transformed and localized sample covariance matrix:

$$\mathbf{B}_e = \mathbf{T} (\mathbf{T}^{-1} \widetilde{\mathbf{B}} \mathbf{T}^{-T} \circ \mathbf{L}) \mathbf{T}^T$$

where:

- $\widetilde{\mathbf{B}} \in \mathbb{R}^{n \times n}$ is the sample covariance matrix estimated from an ensemble,
- $\mathbf{T} \in \mathbb{R}^{n \times n}$ is an invertible transformation matrix,
- $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the localization matrix,
- \circ denotes the Schur product (element-by-element).

- The static covariance model is built with successive parametrized operators:

$$\mathbf{B}_s = \mathbf{U}_b \mathbf{\Sigma} \mathbf{C} \mathbf{U}_b^T$$

where:

- $\mathbf{U}_b \in \mathbb{R}^{n \times n}$ is a multivariate balance operator,
- $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing standard deviations,
- $\mathbf{C} \in \mathbb{R}^{n \times n}$ is a block diagonal (univariate) correlation matrix.

jedi-docs GitHub Repository



GitHub - JCSDA/jedi-docs: JEDI

https://github.com/jcsda/jedi-docs

Why GitHub? Team Enterprise Explore Marketplace Pricing

JCSDA / jedi-docs

Code Pull requests Actions Security Insights

develop 3 branches 0 tags

mmiesch Merge pull request #73 from JCSDA-internal/bugfix/rtd-con... 33902fe 4 days

docs	debugging
.gitattributes	SABER doc - LFS for jpg files
.gitignore	Orion with IntelMPI suite instructions. (#217)
.readthedocs.yml	Fixed up comments
README.md	Add link to JEDI documentation
requirements.txt	Added install of sphinxcontrib-bibtex to the ReadTheDocs conf... 18 days ago

README.md

jedi-docs

This repository is for all [JEDI documentation](#) that doesn't have a logical home in a code repository.

Contributors 15

+ 4 contributors

The JEDI documentation is handled through a GitHub repository just like any of the others

<https://github.com/JCSDA/jedi-docs>

You can fork it, create feature branches, and submit pull requests

jedi-docs GitHub Repository



```
building_jedi.rst
Users > miesch > JEDI > code > local_copy > internal > jedi-docs > docs > using > building_and_running > building_jedi.rst
2
3 Building and compiling JEDI
4 =====
5
6 As described in detail :doc:`elsewhere </>
building and compiling JEDI rests heavily
:code:`ecbuild`, which make your life much
following steps, which are described in m
7
8 1. Clone the desired JEDI :ref:`bundle <bu
9 2. Optionally edit the :code:`CMakeLists.t
you want to work with
10 3. :code:`cd` to the build directory and
other infrastructure
11 4. Run :code:`make update` to pull the lat
:code:`make` to compile the code
12 5. Run :code:`ctest` to verify that the bu
13
14 In terms of the actual commands you would
15
16 .. code-block:: bash
17
18 cd <src-directory>
19 git clone https://github.com/JCSDA/fv3-
20 cd <build-directory>
21 ecbuild <src-directory>/fv3-bundle
22 make update
23 make -i4
```

**Documentation is written as
reStructuredText (rst) files
which are converted to html by the**

Sphinx

Python documentation generator

<https://www.sphinx-doc.org>



SPHINX

Python Documentation Generator

I) The way of a JEDI

- ◆ Agile project management
- ◆ Collaborative
- ◆ Git, GitHub, git-flow

II) Preparing to contribute

- ◆ Work from a fork
- ◆ Make sure your branch is up to date with develop
- ◆ Make sure your code is adequately tested
- ◆ Make sure your code is adequately documented

III) Contributing code

- ◆ Pull requests
- ◆ Code Reviews



Pull Request



The screenshot shows a web browser window displaying the GitHub repository page for 'mmiesch / ufo'. The browser's address bar shows the URL 'https://github.com/mmiesch/ufo/tree/feature/mybranch'. The repository page includes a search bar, navigation tabs for 'Pulls', 'Issues', 'Marketplace', and 'Explore', and a header for the repository 'mmiesch / ufo' with 'Watch', 'Star', and 'Fork' buttons. Below the header, there are tabs for 'Code', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The main content area shows the current branch 'feature/mybran...' and a message indicating it is 55 commits ahead of the 'jcsda-academy:master' branch. A list of recent commits is visible, including 'mmiesch new file' (23 hours ago, 2,201 views), 'CI change jcsda to jcsda-internal (#267)' (10 days ago), and 'cmake removed underflow flag from compiler options' (2 years ago). On the right side, there is an 'About' section with 'Unified Forward Operator', 'Readme', and 'Apache-2.0 License', and a 'Releases' section with '1 tags'.

Pull Request



mmiesch/ufo at feature/mybran... | <https://github.com/mmiesch/ufo/tree/feature/mybranch>

Search or jump to... | Pulls | Issues | Marketplace | Explore

mmiesch / ufo | Watch 0 | Star 0 | Fork 2

forked from [jcsda-academy/ufo](#)

Code | Pull requests | Actions | Projects | Security | Insights | Settings

feature/mybran... | Go to file | Add file | Code

This branch is 55 commits ahead of [jcsda-academy:master](#). | Pull request | Compare

mmiesch	new file ...	23 hours ago	2,201
CI	change jcsda to jcsda-internal (#267)	10 days ago	
cmake	removed underflow flag from compiler options	2 years ago	

About | Unified Forward Operator | Readme | Apache-2.0 License

Releases | 1 tags

Pull Request



mmiesch/ufo at feature/mybran

https://github.com/mmiesch/ufo/tree/feature/mybranch

Search or jump to... Pulls Issues Marketplace Explore

mmiesch / ufo Watch 0 Star 0 Fork 2

forked from jcsda-academy/ufo

Code Pull requests Actions Projects Security Insights Settings

feature/mybran... Go to file Add file Code

This branch is 55 commits ahead of jcsda-academy:master. Pull request Compare

mmiesch new file	23 hours ago	2,201
CI	change jcsda to jcsda-internal (#267)	10 days ago
cmake	removed underflow flag from compiler options	2 years ago

About Unified Forward Operator Readme Apache-2.0 License Releases 1 tags

Pull Request



Comparing JCSDA:develop...miesch:feature/mybranch

https://github.com/JCSDA/ufo/compare/develop...miesch:feature/mybranch?expand=1 133%

Search or jump to... Pulls Issues Marketplace Explore

JCSDA / ufo Watch 25 Star 0 Fork 2


Code Pull requests Actions Security Insights Settings

Open a pull request










Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: JCSDA/ufo base: develop head repository: mmiesch/ufo compare: feature/mybranch


✓ **Able to merge.** These branches can be automatically merged.


 Adding an Operator for the ELVIS instrument

Write Preview

H B I         

Description

Reviewers 
No reviews—at least 2 approving reviews are required.

Assignees 
No one—assign yourself

Pull Request



Comparing JCSDA:develop...mi X

https://github.com/JCSDA/ufo/compare/develop...mmiesch:feature/mybranch?expand=1 133%

Search or jump to... Pulls Issues Marketplace Explore

JCSDA / ufo Watch 25 Star 0 Fork 2

Code Pull requests Actions Security Insights Settings

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: JCSDA/ufo base: develop ← head repository: mmiesch/ufo compare: feature/mybranch

✓ Able to merge. These branches can be automatically merged.

Adding an Operator for the ELVIS instrument

Write Preview

H B I

Description

Reviewers: No reviews—at least 2 approving reviews are required.

Assignees: No one—assign yourself

Pull Request



Comparing JCSDA:develop...mi X

https://github.com/JCSDA/ufo/compare/develop...mmiesch:feature/mybranch?expand=1 133%

Search or jump to... Pulls Issues Marketplace Explore

JCSDA / ufo Watch 25 Star 0 Fork 2

Code Pull requests Actions Security Insights Settings

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: JCSDA/ufo base: develop ← head repository: mmiesch/ufo compare: feature/mybranch

✓ **Able to merge.** These branches can be automatically merged.

Adding an Operator for the ELVIS instrument

Write Preview

H B I

Description

Reviewers
No reviews—at least 2 approving reviews are required.

Assignees
No one—assign yourself

Pull Request



Comparing JCSDA:develop... X +

https://github.com/JCSDA/ufc/compare/develop...mmlesch:feature/mybranch?exp=133%

Adding an Operator for the ELVIS instrument

Write Preview

H B I

Description

(Instructions: this, and all subsequent sections of text should be removed and filled in as appropriate.)
Provide a detailed description of what this PR does.
What bug does it fix, or what feature does it add?
Is a change of answers expected from this PR?

Definition of Done

What does it mean for this issue to be done? Is there a specific, measurable, milestone?

Issue(s) addressed

Link the issues to be closed with this PR
- fixes #<issue_number>

Dependencies

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers

Create pull request

Reviewers
No reviews—at least 2 approving reviews are required.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Use [Closing keywords](#) in the description to automatically close issues

Helpful resources
[GitHub Community Guidelines](#)

**Make it clear
what was
done and
why**

**Refer to
forum
discussions if
applicable**

**JEDI team
will assign
reviewers for
external pull
requests**

Pull Request



Comparing JCSDA:develop... X

https://github.com/JCSDA/ufo/compare/develop...mmlesch:feature/mybranch?exp=133%

Adding an Operator for the ELVIS instrument

Write Preview

H B I

Description

(Instructions: this, and all subsequent sections of text should be removed and filled in as appropriate.)
Provide a detailed description of what this PR does.
What bug does it fix, or what feature does it add?
Is a change of answers expected from this PR?

Definition of Done

What does it mean for this issue to be done? Is there a specific, measurable, milestone?

Issue(s) addressed

Link the issues to be closed with this PR
- fixes #<issue_number>

Dependencies

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers

Create pull request

Reviewers
No reviews—at least 2 approving reviews are required.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Use [Closing keywords](#) in the description to automatically close issues

Helpful resources
[GitHub Community Guidelines](#)

**Make it clear
what was
done and
why**

**Refer to
forum
discussions if
applicable**

**JEDI team
will assign
reviewers for
external pull
requests**

Pull Request



Comparing JCSDA:develop... X

https://github.com/JCSDA/ufc/compare/develop...mmlesch:feature/mybranch?exp=133%

Adding an Operator for the ELVIS instrument

Write Preview

H B I @

Description

(Instructions: this, and all subsequent sections of text should be removed and filled in as appropriate.)

Provide a detailed description of what this PR does.

What bug does it fix, or what feature does it add?

Is a change of answers expected from this PR?

Definition of Done

What does it mean for this issue to be done? Is there a specific, measurable, milestone?

Issue(s) addressed

Link the issues to be closed with this PR

- fix #<issue_number>

Dependencies

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers

Create pull request

**Make it clear
what was
done and
why**

**Refer to
forum
discussions if
applicable**

**JEDI team
will assign
reviewers for
external pull
requests**

Pull Request



Comparing JCSDA:develop... X

https://github.com/JCSDA/ufo/compare/develop...mmlesch:feature/mybranch?exp... 133%

Adding an Operator for the ELVIS instrument

Write Preview

H B I

Description

(Instructions: this, and all subsequent sections of text should be removed and filled in as appropriate.)

Provide a detailed description of what this PR does.

What bug does it fix, or what feature does it add?

Is a change of answers expected from this PR?

Definition of Done

What does it mean for this issue to be done? Is there a specific, measurable, milestone?

Issue(s) addressed

Link the issues to be closed with this PR

- fix #<issue_number>

Dependencies

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers

Create pull request

Reviewers

No reviews—at least 2 approving reviews are required.

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

Use [Closing keywords](#) in the description to automatically close issues

Helpful resources

[GitHub Community Guidelines](#)

Make it clear what was done and why

Refer to forum discussions if applicable

JEDI team will assign reviewers for external pull requests

Pull Request



Adding an Operator for the ELVIS instrument

Write Preview

Description

(Instructions: this, and all subsequent sections of text should be removed and filled in as appropriate.)
Provide a detailed description of what this PR does.
What bug does it fix, or what feature does it add?
Is a change of answers expected from this PR?

Definition of Done

What does it mean for this issue to be done? Is there a specific, measurable, milestone?

Issue(s) addressed

Link the issues to be closed with this PR
- fix #<issue_number>

Dependencies

Attach files by dragging & dropping, selecting or pasting them.

Reviewers
No reviews—at least 2 approving reviews are required.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Use [Closing keywords](#) in the description to automatically close issues

Helpful resources
[GitHub Community Guidelines](#)

Allow edits by maintainers ? [Create pull request](#)

Make it clear what was done and why

Refer to forum discussions if applicable

JEDI team will assign reviewers for external pull requests

Pull Requests



- ▶ **Make feature branches short and focused**
- ▶ **Fill in the requested information in the template**
- ▶ **Explain what was done and why**
- ▶ **What does it mean for this modification to be finished?**
- ▶ **Refer to relevant conversations (forum threads, issues, etc)**
- ▶ **Identify appropriate reviewers**
- ▶ **Make sure new/modified code is tested**
- ▶ **Make sure new/modified code is documented**
- ▶ **Be willing to change your code in response to reviews**
- ▶ **Read the [Working Principles](#) and [Best Practices for Developers](#) sections of the JEDI Documentation**

Code Reviews



Purpose

To ensure that the overall health of the code (scope, functionality, clarity, efficiency, reliability) improves over time

Requirements

To be useful, they must be timely, courteous, informative, constructive, and reasonable (there is no perfect code, only better code)

Additional Benefits

Sharing knowledge, team building and mentoring,
improving the development process,
imposing a consistent style & coding norms

Questions to ask yourself as a reviewer



Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***

Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***
- ▶ ***Is it clear from the title and description what is being done and why? Does it achieve what it says it does?***

Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***
- ▶ ***Is it clear from the title and description what is being done and why? Does it achieve what it says it does?***
- ▶ ***Can the desired goal be achieved in a different way that is more readable, more efficient, or more generic?***

Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***
- ▶ ***Is it clear from the title and description what is being done and why? Does it achieve what it says it does?***
- ▶ ***Can the desired goal be achieved in a different way that is more readable, more efficient, or more generic?***
- ▶ ***Is there extraneous code that should be removed (e.g. debug print statements, unnecessary include statements...)?***

Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***
- ▶ ***Is it clear from the title and description what is being done and why? Does it achieve what it says it does?***
- ▶ ***Can the desired goal be achieved in a different way that is more readable, more efficient, or more generic?***
- ▶ ***Is there extraneous code that should be removed (e.g. debug print statements, unnecessary include statements...)?***
- ▶ ***Is the new code adequately tested? Does it pass all tests?***

Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***
- ▶ ***Is it clear from the title and description what is being done and why? Does it achieve what it says it does?***
- ▶ ***Can the desired goal be achieved in a different way that is more readable, more efficient, or more generic?***
- ▶ ***Is there extraneous code that should be removed (e.g. debug print statements, unnecessary include statements...)?***
- ▶ ***Is the new code adequately tested? Does it pass all tests?***
- ▶ ***Is the new code adequately documented?***

Questions to ask yourself as a reviewer



- ▶ ***Does this improve the overall health of the code?***
- ▶ ***Is it clear from the title and description what is being done and why? Does it achieve what it says it does?***
- ▶ ***Can the desired goal be achieved in a different way that is more readable, more efficient, or more generic?***
- ▶ ***Is there extraneous code that should be removed (e.g. debug print statements, unnecessary include statements...)?***
- ▶ ***Is the new code adequately tested? Does it pass all tests?***
- ▶ ***Is the new code adequately documented?***
- ▶ ***Does this belong in the code base or elsewhere (e.g. library)?***

Questions to ask yourself as a reviewer



- ▶ **Does this improve the overall health of the code?**
- ▶ **Is it clear from the title and description what is being done and why? Does it achieve what it says it does?**
- ▶ **Can the desired goal be achieved in a different way that is more readable, more efficient, or more generic?**
- ▶ **Is there extraneous code that should be removed (e.g. debug print statements, unnecessary include statements...)?**
- ▶ **Is the new code adequately tested? Does it pass all tests?**
- ▶ **Is the new code adequately documented?**
- ▶ **Does this belong in the code base or elsewhere (e.g. library)?**
- ▶ **Have I read the [Working Principles](#) and [Best Practices for Developers](#) sections of the JEDI Documentation?**



Working Principles — JEDI Doc x +

jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/latest/working-practices/index.html

Apps Washington Post:... Github-JCSDA Da... Teams · JCSDA JEDI1 - JEDI JEDI1 · Infrastruct... EPIC JEDI Documentati... Workday JEDI

Search docs

Overview

Working Principles

- Branching and merging code
- Forking and cloning repositories
- Reviewing Code
- Testing
- Creating documentation

Learning JEDI

Using JEDI

Inside JEDI

Frequently Asked Questions (FAQ)

References

Read the Docs v: latest

Working Principles

- [Branching and merging code](#)
- [Forking and cloning repositories](#)
- [Reviewing Code](#)
 - [What is a Code Review?](#)
 - [Creating a Good Pull Request](#)
 - [The Standard of Code Review](#)
 - [Benefits of Code Reviews](#)
 - [What to look for in a Code Review](#)
 - [How Fast Should Code Reviews Be?](#)
 - [Comments in a code review](#)
 - [Give and Take in a Code Review](#)
- [Testing](#)
- [Creating documentation](#)

Previous Next



Best Practices for Developers · × +

jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/latest/inside/practices/index.html

Apps Washington Post:... Github-JCSDA Da... Teams · JCSDA JEDI1 - JEDI JEDI1 · Infrastruct... EPIC JEDI Documentati... Workday JEDI

Using JEDI

- Inside JEDI
 - JEDI Components
 - JEDI Testing
 - Best Practices for Developers
 - Create PLEATED Issues to let your team know what you are working on
 - Follow the Git flow Paradigm
 - TRIPLE the impact of your GitHub Pull Requests
 - Document your code
 - Treat ECMWF Forks as Forks
 - Developer Tools
 - Frequently Asked Questions (FAQ)
 - References

Read the Docs v: latest ▾

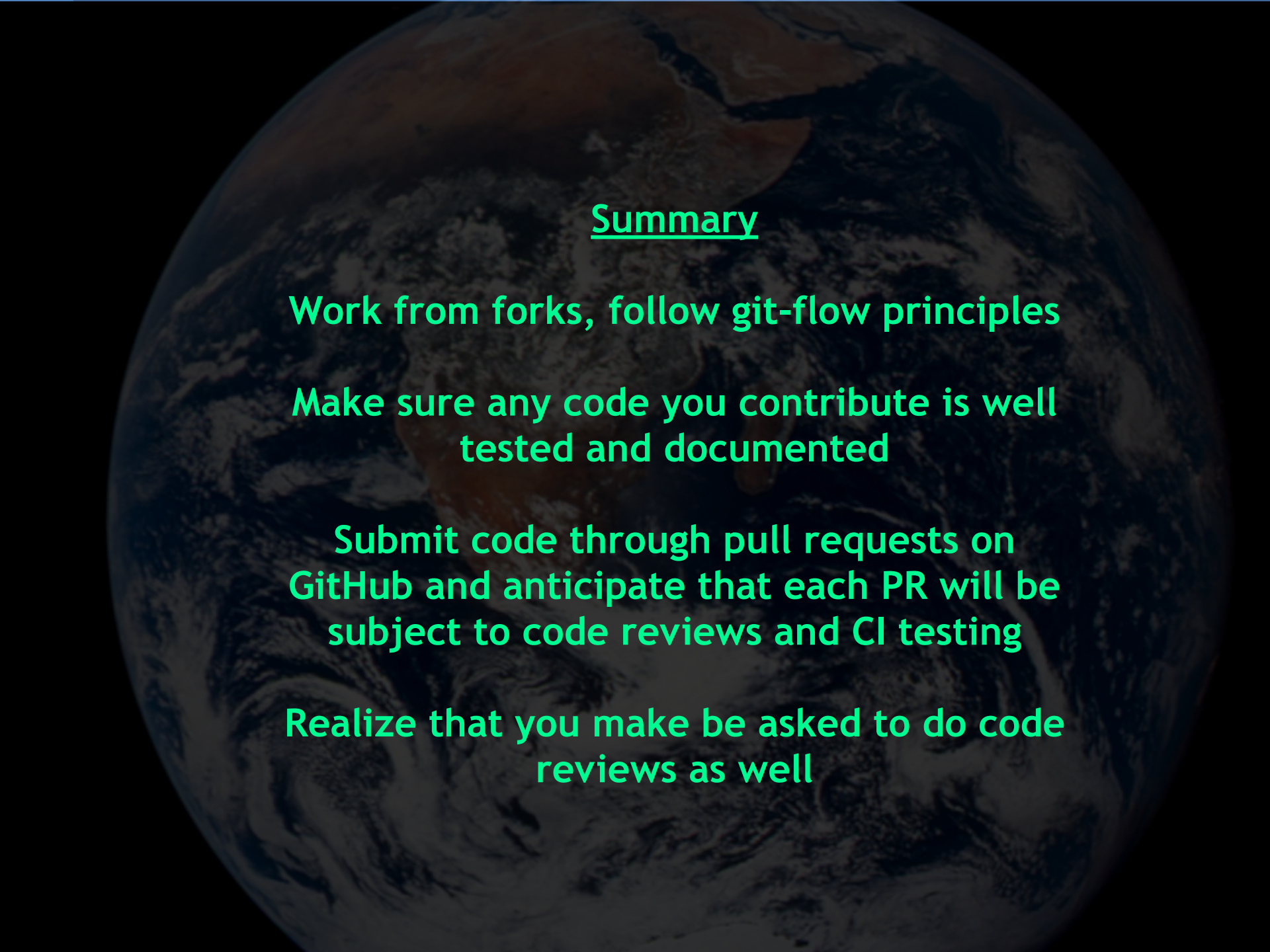
Docs » Inside JEDI » Best Practices for Developers

[Edit on GitHub](#)

Best Practices for Developers

- Create PLEATED Issues to let your team know what you are working on
- Follow the Git flow Paradigm
 - Life Cycle of a Feature Branch
- TRIPLE the impact of your GitHub Pull Requests
- Document your code
- Treat ECMWF Forks as Forks

[← Previous](#) [Next →](#)



Summary

Work from forks, follow git-flow principles

Make sure any code you contribute is well tested and documented

Submit code through pull requests on GitHub and anticipate that each PR will be subject to code reviews and CI testing

Realize that you may be asked to do code reviews as well