



## **Lessons learned**

Ibrahim Haddad, Ph.D.

VP Strategic Programs, Linux Foundation

Executive Director, LF AI Foundation

This deck was contributed by Ibrahim Haddad to LF Energy and is licensed under CC BY 4.0.



@IbrahimAtLinux



IbrahimAtLinux.com



linkedin.com/in/ibrahimhaddad/



Ibrahim@Linux.com



github.com/ibrahimhaddad

**LF**ENERGY

# Learned Lessons - Internally

*Listed in random order*

- › Policies, procedures, training, education and a central open source org
- › Establish and execute an open source strategy with a long term view
- › Work with open source model internally before you do it externally
- › Define your goals – what do you want to achieve
- › Increase participation in open standards
- › Increase participation in open source foundations
- › Adopt an upstream first philosophy

## Learned Lessons - Externally

- › Work with the community: Provide contributions to community and work with community to integrate with mainline
- › Clarify license issues with released code
- › "Scratch your own itch"
- › Participate and start new open source projects
- › Visibility is important
- › Avoid forking, branching, fragmentation, back-porting
- › Work with open source foundations and open standards
- › Document changes to the code

# Participating in open source - Looking for leverage

- › Expect to “scratch your own itch”
  - › Don't expect others to scratch it for you.
  - › Leverage happens when others share the same itch
  - › Faster results or lower cost or requiring fewer skills
- › If no one shares the “itch”:
  - › You can still work your itch (usually, not always) into a community solution
  - › You'll usually have to bear the whole cost or effort
- › Hardest way to leverage open source is to start a community from scratch
  - › Find a like-minded community with your solution in mind, join to accelerate
  - › Find an adjacent community, join to help broaden, adapt their solution
  - › Build a new community of common-minded partners

# Advice on Participating in Open Source

- › Listen to Open Source developers
  - › they know their stuff, specially regarding integration and distribution
- › Keep an eye and participate on user forums, mailing lists, IRC, Slack, Glitter, etc.
  - › they use it everyday
  - › listen to their reports, monitor and participate in the mailing lists
- › Be open
  - › Disclose problems fast, along with immediate workarounds and early fixes
  - › Pretending everything is right will buy you trouble
- › Reuse code
- › Release early and often
- › Gives a good estimate on progress, helps catching bugs early
- › Don't be perfect
- › Ask around in mailing lists or public forums
  - › Let it be known that your company is interested in finding *The Right Way* to do things
- › Use best known methods, and follow community processes
- › If you find a better one, make sure it becomes *best known*

## What to adopt from oss development model?

- › Team communication - Use mailing lists for project discussions
- › User involvement
- › Peer review
- › Reuse - Attempt to re-use existing open source components
- › Build reusable software components
- › Pay special attention to quality and security
- › Release early and often - Apply small incremental changes in the release to make it easier to understand and test
- › Transparency

# What to adopt from oss development model?

- › Respect and follow coding style
- › Flag problems/bugs early & review with team
- › Make source code available for others to review
- › Staffing methods , Core and non-core developers
- › Continuous integration and automated test environments
- › Minimal code base and new functionalities in separate modules



# Dan Frye, IBM VP of Open Systems, 10/2/2007

IBM Open Systems Development

## Other things I've learned in 8 years of Open Source Development

- ✓ *Do initiate open source projects early*
- ✗ *Don't try to throw completed code "over the wall"*
- ✓ *Do manage your OSS developers*
- ✗ *Don't try to manage their maintainers*
- ✓ *Do listen to and participate in IRC chat*
- ✗ *Don't pick a public fight with people who have their own channel, blog, wiki, and more*
- ✗ *Don't expect everyone to embrace your specialized approach or function*
- ✓ *Do welcome help in crafting a more general solution*
- ✗ *Don't reject criticism or even counter-attack*
- ✓ *Do use the opportunity to engage in discussion to refine the goal or improve the solution*
- ✗ *You do not control the open source process*
- ✓ *You can influence the open source community, especially when you've established expertise and willingness to contribute*
- ✓ *Do be persistent*
- ✗ *Don't be stubborn*

*Open Source is amazingly, even surprisingly, effective*

# How can we make it work?

1. Create the right environment
2. Provide the right incentive
3. Provide the right exposure
4. Integrate your teams within the specific projects' communities



## Lessons learned

Ibrahim Haddad, Ph.D.

VP Strategic Programs, Linux Foundation

Executive Director, LF AI Foundation